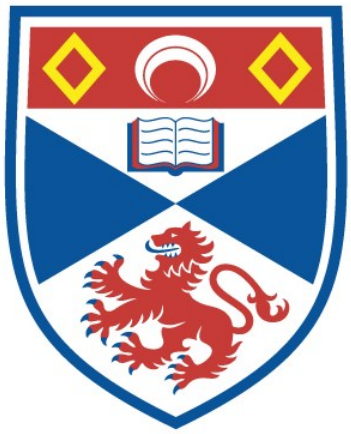
COMMUTATIVITY AND FREE PRODUCTS IN THOMPSON'S GROUP *V*

Ewa Bieniecka

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



2018

Commutativity and Free Products in Thompson's Group V

Ewa Bieniecka

University of
St Andrews

This thesis is submitted in partial fulfilment
for the degree of PhD at the University of St Andrews

10th November, 2017

Declaration

1. Candidate's declarations:

   I, Ewa Bieniecka, hereby certify that this thesis, which is approximately 50,000 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for a higher degree.

   I was admitted as a research student in January 2013 and as a candidate for the degree of PhD in January 2013; the higher study for which this is a record was carried out in the University of St Andrews between 2013 and 2017.

   Date ..... signature of candidate .....

2. Supervisor's declaration:

   I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Ph.D. in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

   Date ..... signature of supervisor .....

   Date ..... signature of supervisor .....

3. Permission for publication:

   In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research

worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. I have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

PRINTED COPY: a) No embargo on print copy.

ELECTRONIC COPY: a) No embargo on electronic copy.

Date ..... signature of candidate .....

Date ..... signature of supervisor .....

Date ..... signature of supervisor .....

Abstract

We broaden the theory of dynamical interpretation, investigate the property of commutativity and explore the subject of subgroups forming free products in Thompson's group V.

We expand Brin's terminology [11] for a revealing pair to an any tree pair. We use it to analyse the dynamical behaviour of an arbitrary tree pair which cannot occur in a revealing pair. Hence, we design a series of algorithms generating Brin's revealing pair [11] from any tree pair, by successively eliminating the undesirable structures. To detect patterns and transitioning between tree pairs, we introduce a new combinatorial object called the *chains graph*. A newly defined, unique and symmetrical type of a tree pair, called a *balanced tree pair*, stems from the use of the chains graphs.

The main theorem of Bleak *et al.* [5] states the necessary structure of the centraliser of an element of V. We provide a converse to this theorem, by proving that each of the predicted structures is realisable. Hence we obtain a complete classification of centralisers in V. We give an explicit construction of an element of V with prescribed centraliser. The underlying concept is to embed a Cayley graph of a finite group into the *flow graph* (introduced in Bleak *et al.* [5]) of the desired element. To reflect the symmetry, we present the resulting element in terms of a balanced tree pair.

The group V is conjectured to be a universal coCF group, which generates interest in studying its subgroups. We develop a better understanding of embeddings into V by providing a necessary and sufficient dynamical condition for two subgroups (not both torsion) to form a free product in V. For this, we use the properties, explored in Bleak and Salazar–Díaz [8], of sets of so–called *important points*, and the Ping–Pong action induced on them.

Acknowledgements

Many thanks to Dr Collin Bleak, my research advisor, for all the fruitful discussions about Thompson's group V, and for his support during my transition to PhD program; to Dr Martyn Quick, my research advisor, for sharing his wisdom and expertise in writing mathematics; to my colleague Hunter Spink for helpful conversations about automorphisms of graphs; to Andrew Duncan and David Robertson for conversations about flow graphs and conjugacy in V; and also to Fenrir Thorvaldsen for always being there for me.

The pictures of tree pairs are taken from the *vTrees* software package written by Collin Bleak and Roman Kogan [6].

Contents

Chapter 1

Introduction

In this thesis we expand the theory of dynamical interpretation, investigate the property of commutativity and explore the subject of subgroups forming free products in Thompson's group V. The family of Thompson's groups, consisting of three groups currently known as F, T and V, have been introduced in the mid-1960s by Richard J. Thompson in [31]. The group F was proposed as a possible candidate for an answer to a very well–known problem in logic, while T and V were the first known examples of finitely presented infinite simple groups.

1.1 History

In 1924 two Polish mathematicians published a paper which shocked the public by questioning the paradigms of intuitive understanding of a concept as fundamental as volume. Understandably, the Banach–Tarski paradox [1], presenting a paradoxical decomposition of a sphere using a particular group action on it, drew the attention of a lot of their contemporary researchers. In 1929 von Neumann [26] first defined the concept of amenability for groups, and Tarski [29], [30] proved that a paradoxical decomposition of the sphere can occur if and only if the group acting on it is non–amenable. However, actual constructions seemed to rely on the usage of a non–abelian free subgroup. Von Neumann [26] showed that existence of a non–abelian free subgroup implies non–amenability. Henceforth, a natural conjecture arose, which speculated whether non–amenable groups are precisely those which admit non–abelian free subgroups. In the mid-1950s the name of von Neumann was explicitly linked to this conjecture. In 1965, when the problem remained both unsolved and increasingly pop-

ular, Richard Thompson [31] introduced the first potential counterexample to it, which we currently know as Thompson's group F. The von Neumann conjecture was disproved in 1980 by Ol'shanski [27], but F remained a candidate for the first finitely–presented counterexample to the conjecture. Only in 2013, Monod [25] provides a very natural counterexample, which in fact is related to F, and in 2016 Lodha and Moore [23] find a subgroup of this counterexample which is a non–amenable finitely–presented group with no non–abelian free subgroups. Interestingly, the problem whether F is amenable or not remains open to this day.

The Thompson's group F remains the most famous of the groups introduced by Thompson in [31]. However, he simultaneously defined two other groups, currently known as T and V, which naturally generalise F and satisfy the containments $F \leq T \leq V$. The Thompson's groups F, T and V possess a set of unusual properties which make them natural candidates for counterexamples in various conjectures in group theory. The groups T and V are for that matter instances of infinite, finitely–presented simple groups. In particular, the motivation for this work has connections to the co–word problem, which will be explained later in more detail.

Over the years, we also observe many new families of groups inspired by Thompson's groups. Amongst most famous we have Higman–Thompson family of groups $G_{n,r}$ investigated in 1974 by Higman in [17], groups of piecewise–linear homomorphisms of the real line analysed in 1985 by Brin and Squier in [12] and a family of groups of the form nV introduced in 2004 by Brin in [11].

1.2 How V Has Been Studied

There are many approaches which have been taken in order to study Thompson's groups. They can be described using presentations, as for instance in Cannon, Floyd and Parry [14], which often serves as a first point of contact with these groups. In some recent studies, Bleak and Quick [9] point out the resemblance of a presentation of V with finite symmetric and alternating groups. In Higman [17] the interpretation of V occurs via its action on an algebra, which approach was continued recently by Salazar–Díaz in [28]. Cannon, Floyd and Parry [14] also describe how an element of V can be understood as a (non–unique) combinatorial object known as a tree pair. This is possibly the most widely

used strategy, although it is often accompanied by other techniques. As an example, it has been used in recent studies of Matucci [24] to construct so called strand diagrams. In Bleak and Salazar–Díaz in [8], Thompson's group V is understood as a group of homeomorphisms of Cantor space. This approach is continued and expanded by a new dynamical tool called a flow graph in Bleak *et al.* in [5].

In this thesis, the group V is viewed as a group of bijections on Cantor space and understood via this action. However, elements of V are also depicted by tree pairs, and represented by new objects introduced in this thesis, called chains graphs. We will describe our methods in more detail in Section 1.4.

1.3 Motivation for Studying Embeddings into V

A context–free (CF) group is a group in which all expressions equal to the identity form a CF language. The syntax of most widely used languages in computer programming is CF (or almost–CF), because CF parsing is well–studied, with efficient algorithms. A co–context–free (coCF) group is a group in which the complement of all expressions equal to identity is a CF language. Lehnert and Schweitzer show in [22] that V is a coCF group, which implies that all of its finitely generated subgroups are also coCF. There is a possibility that there exists a universal coCF group, namely a group which contains all coCF groups as its subgroups. Lehnert [21] proposes a candidate for a universal coCF group. Bleak, Matucci and Neunhöffer in [7] obtain an equivalent version of Lehnert's conjecture, which suggest V as a universal coCF group. Because of this conjecture, and because all finitely generated groups of V are coCF, the study of embeddings into V is of particular interest. This provides motivation for the research presented in Chapter 5, where we analyse dynamical conditions required for free product embeddings.

1.4 Description of Our Methods

As mentioned before, V is understood as a group of homeomorphisms on Cantor space. Studying this action of V enabled Bleak and Salazar–Díaz in [8] to decide that the group $\mathbb{Z}^2 * \mathbb{Z}$ does not embed into V. Simultaneous studies and the development of a new combinatorial tool for analysing this action enabled a thorough analysis of allowable isomorphism type of

centralisers of elements of group V by Bleak *et al.* in [5]. In Chapter 4 of this thesis, I produce a converse to their main result, which results in Theorem 4.1.2. My method focuses on constructing elements of Thompson's group V with prescribed isomorphism type of their centraliser, realising all types predicted by the model from [5].

The dynamical tool developed by Bleak *et al.* in [5], known as a flow graph, is derived from the concept of a special kind of a tree pair, described by Brin in [11] and called a revealing pair. This type of tree pair is useful for understanding the dynamics occurring on Cantor space under the action of V. In [28] Salazar–Díaz presented a method to find all revealing pairs corresponding to a given element $\alpha \in V$, given any of its revealing pairs. Interestingly, no formally written algorithm for obtaining a revealing tree pair from a non–revealing tree pair was published, even though the idea was known to be discussed by Collin Bleak and Daniel Farley. Chapter 3 of this thesis presents my work obtaining the desired algorithm. Moreover, in the process of analysing the problem, I use an object inspired by a flow graph which I call a chains graph. The chains graph allows for dynamical analysis of any tree pair, not only a revealing tree pair. It helps to understand all possible behaviours which might occur in a tree pair (see Definition 3.1.27 and Corollary 3.1.29) and hence allows for constructing an algorithm which eliminates behaviours unacceptable for a revealing tree pair. In addition, we find types of tree pairs which are related to revealing pairs, but unlike them are unique. We call them balanced tree pairs. They are implicitly used in Construction 4.3.13 in Chapter 4 as more symmetric alternatives to revealing pairs.

Finally, relating back to embedding and non–embedding results, as an example of an open question, it is not known whether surface groups embed into V. Because of the presentation of surface groups, it seems that understanding of how free products embed into V, and in particular what effect they have on Cantor space, would be useful for targeting this problem. There is also a related open question posed by Bleak and Salazar–Díaz in [8], whether for any free product in V we need to have a Ping–Pong dynamics on Cantor space. We know that Ping–Pong dynamics implies decomposition of the acting group as a free product, but the converse doesn't hold in general. In Chapter 5, I present my contribution to a joint project with Bleak and Matucci on the subject of this open question. In particular, my Theorem 5.1.1 states that if two subgroups of V canonically form a free product, and if there is at least one element

of infinite order contained in one of them, then the groups admit a Ping–Pong dynamics on a set of subsets of Cantor space. In order to prove this theorem, I use techniques involving so called important points and apply results from Bleak and Salazar–Díaz [8]. The joint project is in progress of expanding this result to generalise to free products consisting entirely of torsion, and also for finding Ping–Pong dynamics directly on a set of points from Cantor space. One of the ideas is to apply the construction used in Bennett and Bleak in [2] of Ping–Pong dynamics for demonstrative groups.

Chapter 2

Fundamentals of Thompson's Group V

In this chapter, we will introduce standard concepts related to Thompson's group V, including its representation as a collection of tree pairs. Sections 2.1 and 2.2 are based on exposition given in Bleak and Salazar–Díaz [8], and in Bennett and Bleak [2], but expanded to fit our needs.

2.1 Cantor Space

Let $X = \{0, 1\}$. The set X will be called our alphabet, and the members of X will be called letters. Let X^* denote the set of all finite strings over X, including the empty string ϵ, and let X^+ denote the set of all finite non–empty strings over X. The elements of X^* will be called *words*. Note that elements of X^* may be viewed as elements of a monoid on the set of generators X, where the product of two words is given by their concatenation.

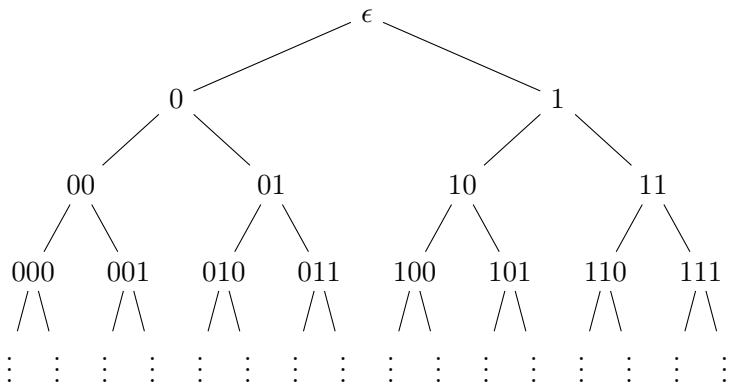
Define the infinite rooted binary tree $\mathcal{T}$ as a non–directed graph with the following properties:

1. The vertex set of $\mathcal{T}$ is given by X^*.
2. The root of $\mathcal{T}$ is given by the empty word ϵ.
3. For any words $u, v \in X^*$, there is an edge between u and v, if and only if, $us = v$ or $vs = u$ for some letter $s \in X$.

A vertex of $\mathcal{T}$ will also be referred to as a *node* or an *address*. Suppose that $u, v \in X^*$. If there is a letter $s \in X$ such that $us = v$, then we say

that u *is a parent of* v and that v *is a child of* u. More generally, if there is a word $w \in X^*$ such that $uw = v$, then we say that u *is an ancestor of* v and that v *is a descendant of* u. In this case we also say that u *is a prefix of* v.

We can imagine the tree $\mathcal{T}$ in the following way. Its root ϵ is at the top. It has two children, 0 and 1, both pictured to be situated a level lower than ϵ. The child 0 is on the left side, and the child 1 is on the right side. In general, given a node u of $\mathcal{T}$, we draw its child $u0$ a level below and to the left, and we draw its child $u1$ a level below and to the right.



The *boundary of* $\mathcal{T}$ is the set of all infinite strings over the alphabet X, denoted as $\{0,1\}^\omega$. We will also refer to them as *infinite words*. We now identify the standard Cantor set $\mathfrak{C}$ with the boundary of $\mathcal{T}$.

We will extend the notion of a prefix to the following situation: suppose that $p \in \{0,1\}^\omega$ is an infinite word, and $w \in X^*$ is a (finite) word. If there exists an infinite word $r \in \{0,1\}^\omega$ such that $wr = p$, then we say that w *is a prefix of* p. We also say that p *underlies the node* w.

2.2 Prefix Substitution

As mentioned before, the group V may be viewed as a specific collection of automorphisms of Cantor space, under the operation of composition. Before we define V, we will introduce the notation relating to the action of V on $\mathfrak{C}$.

First of all, we will assume that V acts on the Cantor set on the right. Hence, for any $p \in \mathfrak{C}$ and any $\alpha \in V$, we will denote $p\alpha$ as the image of p under the application of α. This also implies that by convention, for

$\alpha, \beta \in V$, conjugation is given by $\alpha^\beta = \beta^{-1}\alpha\beta$ and commutation is given by $[\alpha, \beta] = \alpha^{-1}\beta^{-1}\alpha\beta$. Furthermore, we will define the *support of* $\alpha \in V$ by $\mathrm{Supp}(\alpha) = \{p \in \mathfrak{C} \mid p\alpha \neq p\}$. We will also define the orbit of p under the action of $\langle\alpha\rangle$ by $\mathcal{O}_{(p,\alpha)} = \{p\alpha^k \mid k \in \mathbb{Z}\}$. If this set is finite, we say that the orbit is *periodic*. If this set has only one element, then we say that the orbit is *trivial*.

We will now focus on the group V. Consider a finite non–empty subset S of m words from X^*, such that the following holds: for each point p in $\mathfrak{C} = \{0,1\}^\omega$ there is a unique word w in S, such that w is a prefix of p. Note that $S = \{\epsilon\}$ is an acceptable choice of a set with one element, as ϵ is a prefix for any point in $\mathfrak{C}$. As Holt and Röver in [18], we call such a set a *barrier*. Another example of a barrier is set $\{00, 01, 1\}$. The set $\{0, 00, 11\}$, however, is not a barrier, as points in $\mathfrak{C}$ starting with prefix 00 admit two prefixes from this set, while points starting with prefix 10 admit no prefixes from this set. Notice that a barrier has a natural lexicographic order inherited from Cantor space. For instance, the elements of the set $\{00, 01, 1\}$ are ordered as follows: $00 < 01 < 1$.

Definition 2.2.1. The elements of Thompson's group V are all maps on $\mathfrak{C}$ under the operation of composition, which can be defined as follows:

1. Fix a positive integer m.

2. Pick two barriers of size m with their elements in the lexicographic order $\mathcal{D} = \{u_1, u_2, \ldots u_m\}$ and $\mathcal{R} = \{v_1, v_2, \ldots, v_m\}$.

3. Pick an m-tuple $t = (t_1, t_2, \ldots, t_m)$ representing a permutation of the set $\{1, 2, \ldots, m\}$.

4. An element $\alpha \in V$ represented by the triple $(\mathcal{D}, \mathcal{R}, t)$ is given by the following map on $\mathfrak{C}$: for all $w \in \{0,1\}^\omega$ and for all i such that $1 \leq i \leq m$ we define $(u_{t_i}w)\alpha = v_i w$.

Let us consider an example of an element of the group V defined as above:

Example 2.2.2. Let $m = 3$, the barrier $\mathcal{D} = \{00, 01, 1\}$ and the barrier $\mathcal{R} = \{0, 10, 11\}$. We observe that $00 < 01 < 1$ and hence $u_1 = 00$, $u_2 = 01$ and $u_3 = 1$. Similarly, we observe that $0 < 10 < 11$ and hence $v_1 = 0$, $v_2 = 10$ and $v_3 = 11$. Let $t = (2, 3, 1)$. We conclude that the map

α is given by:

$$\begin{array}{lll} (u_{t_1}w)\alpha = (u_2w)\alpha = v_1w & \text{so} & (01w)\alpha = 0w \\ (u_{t_2}w)\alpha = (u_3w)\alpha = v_2w & \text{so} & (1w)\alpha = 10w \\ (u_{t_3}w)\alpha = (u_1w)\alpha = v_3w & \text{so} & (00w)\alpha = 11w \end{array}$$

for all $w \in \{0,1\}^\omega$.

Note that the representation of α by a triple is not unique. For instance the identity map might be represented by both $(\{\epsilon\},\{\epsilon\},(1))$ and $(\{0,1\},\{0,1\},(1,2))$.

We refer to this type of map as *prefix substitution*. Note that the prefix substitution map α can be extended to all elements of X^* which have a (not necessarily proper) prefix in the barrier $\mathcal{D}$. Therefore, α induces a partial action on almost all of the nodes of $\mathcal{T}$. More precisely, for all words $w \in X^*$ and for all i such that $1 \leq i \leq m$, we have $(u_{t_i}w)\alpha = v_iw$.

We argue that the map α represented by a triple $(\mathcal{D},\mathcal{R},t)$ is a bijection on $\mathfrak{C}$. As $\mathcal{D}$ is a barrier, the map α is well–defined on each point of $\mathfrak{C}$. As each element of $\mathfrak{C}$ has a prefix in $\mathcal{R}$, α maps onto $\mathfrak{C}$. As this prefix is unique and because $\mathcal{D}$ is a barrier, the map α is also one–to–one. Hence $V \subseteq \mathrm{Sym}(\mathfrak{C})$.

As the map α is a bijection on $\mathfrak{C}$, it is invertible in $\mathrm{Sym}(\mathfrak{C})$. We will show that its inverse α^{-1} belongs to V.

Lemma 2.2.3. *If $\alpha \in V$ then $\alpha^{-1} \in V$.*

Proof. Let $(\mathcal{D},\mathcal{R},t)$ be a triple representing an element α of Thompson's group V. Let m be the order of the barriers $\mathcal{D}$ and $\mathcal{R}$. Let the elements of the barriers be given in the lexicographic order by $\mathcal{D} = \{u_1,u_2,\ldots u_m\}$ and $\mathcal{R} = \{v_1,v_2,\ldots,v_m\}$. We know that the bijection α on $\mathfrak{C}$ is given by the map:

$$(u_{t_i})\alpha = v_i$$

for all i such that $1 \leq i \leq m$. Hence, the map α^{-1} needs to be given by:

$$(v_i)\alpha^{-1} = u_{t_i}$$

for all i such that $1 \leq i \leq m$. Consider the ordering of the set $\{t_1,t_2,\ldots,t_m\}$

to be

$$t_{j_1} = 1 < t_{j_2} = 2 < \ldots < t_{j_m} = m$$

for $j_1, j_2, \ldots, j_m \in \{1, 2, \ldots, m\}$.

Define $t' = (t'_1, t'_2, \ldots, t'_m) = (j_1, j_2, \ldots, j_m)$. Hence we may rephrase the rule for α^{-1} as:

$$(v_{j_i})\alpha^{-1} = u_{t_{j_i}} \qquad \text{which implies that} \qquad (v_{t'_i})\alpha^{-1} = u_i$$

for all i such that $1 \leq i \leq m$.

Thus, the triple $(\mathcal{R}, \mathcal{D}, t')$ represents the element α^{-1} which confirms that $\alpha^{-1} \in V$. □

We will prove in Lemma 2.3.10, when we develop our understanding of equivalent tree pairs, that the composition of two elements of V is an element of V. This is the last property we need, to show that V is a group.

2.3 Tree Pairs

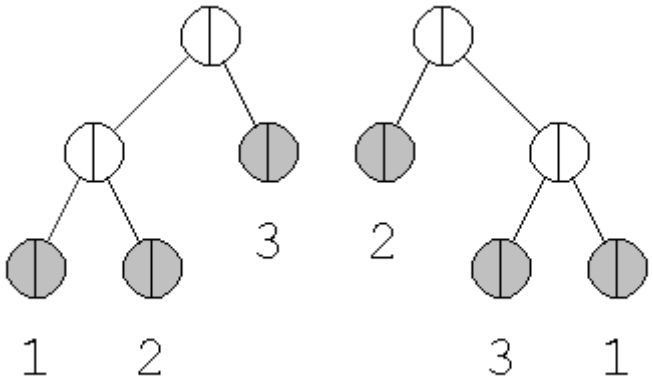
Elements of V as Tree Pairs

We will describe a very popular way in which elements of Thompson's group V are presented.

Definition 2.3.1 (Tree Pair)**.** Let m be a positive integer, and let $\mathcal{D}$ and $\mathcal{R}$ be two barriers of order m. Let D and R both be finite subtrees of $\mathcal{T}$, such that D has the barrier $\mathcal{D}$ as its set of leaves and R has the barrier $\mathcal{R}$ as its set of leaves. Let t be an m-tuple representing an arrangement of numbers from the set $\{1, 2, \ldots, m\}$. Note that $(\mathcal{D}, \mathcal{R}, t)$ is an element of Thompson's group V. We define a *tree pair* to be a triple (D, R, t).

A tree pair (D, R, t) representing an element $\alpha \in V$ is typically depicted by the trees D and R with their leaves decorated. We label the leaves of the tree D with numbers from 1 to m, from left to right. For the m-tuple $t = (t_1, t_2, \ldots, t_m)$, we label the leaves of R with numbers from t_1 to t_m, from left to right.

Example 2.3.2. Let us see how the element of V from Example 2.2.2 would look as a tree pair. Recall that $m = 3$, the barrier $\mathcal{D} = \{00, 01, 1\}$, the barrier $\mathcal{R} = \{0, 10, 11\}$ and $t = (2, 3, 1)$. The tree pair looks as follows:



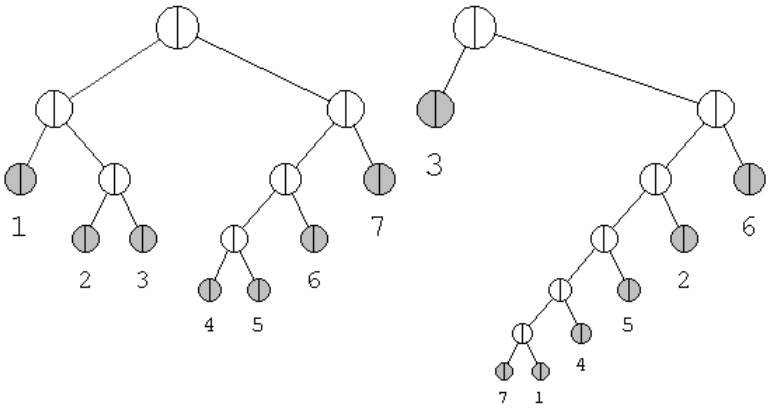
This visual representation makes it intuitive what the prefix substitution for α is: for each $i \in \{1, \ldots, m\}$ the leaf u of D labelled with the number i is mapped to the leaf v of R labelled with $t_j = i$ for some $j \in \{1, \ldots, m\}$.

We will show that this is indeed the same rule as given in Definition 2.2.1 of elements of the group V. Let us consider all leaves of D in their lexicographic order, namely $u_1 < u_2 < \ldots < u_m$ for $\mathcal{D} = \{u_1, u_2, \ldots, u_m\}$. Note that for all $i \in \{1, \ldots, m\}$, the leaf u_i is decorated with the number i. Now let us consider all leaves of R in their lexicographic order, namely $v_1 < v_2 < \ldots < v_m$ for $\mathcal{R} = \{v_1, v_2, \ldots, v_m\}$. Note that for all $i \in \{1, \ldots, m\}$, the leaf v_i is decorated with the number t_i. Hence, for all $i \in \{1, \ldots, m\}$, both u_{t_i} of the tree D and v_i of the tree R are decorated with number t_i. Hence, the map α is given by: for all $w \in \{0, 1\}^\omega$ we have $(u_{t_i} w)\alpha = v_i w$. This is precisely the rule with which we defined elements of V.

A tree pair defined and understood in this way will then always represent an element of V. Also, by definition, each element of V can be represented by (many in fact!) tree pairs. See Algorithms 2.3.9 and 3.2.28 for a description of methods of creating other tree pairs representing α from a given tree pair corresponding to α.

Let us have a look at an example of a tree pair and the corresponding bijection on the Cantor space:

Example 2.3.3 (Example of a Tree Pair)**.** Consider a tree pair (D, R, t) represented by the tree pair below.



The set of all leaves of D is given by

$$\mathcal{D} = \{00, 010, 011, 1000, 1001, 101, 11\}.$$

Also, the set of all leaves of R is given by

$$\mathcal{R} = \{0, 100000, 100001, 10001, 1001, 101, 11\}$$

The 7-tuple t is given by $(3, 7, 1, 4, 5, 2, 6)$. Hence, the tree pair (D, R, t) corresponds to an element $(\mathcal{D}, \mathcal{R}, t)$ of Thompson's group V.

We will describe how this tree pair translates into a bijection on the Cantor space via prefix substitution.

For all $w \in \{0, 1\}^\omega$ we have:

$$\begin{aligned}
(00w)\alpha &= 100001w \\
(010w)\alpha &= 101w \\
(011w)\alpha &= 0w \\
(1000w)\alpha &= 10001w \\
(1001w)\alpha &= 1001w \\
(101w)\alpha &= 11w \\
(11w)\alpha &= 100000w
\end{aligned}$$

Notation for Tree Pairs

Notice that α as in Example 2.3.3 might be interpreted as mapping leaves of the tree D to the leaves of the tree R. Therefore, we give the following names for these trees:

Definition 2.3.4 (Domain Tree and Range Tree)**.** Consider a tree pair (D, R, t). We call the tree D the *domain tree* of this tree pair, and we call the tree R the *range tree* of this tree pair.

As we will be interpreting elements of V as tree pairs, it will be convenient to have a notation for barriers as sets of leaves of tree pairs.

Definition 2.3.5 (Leaf Set)**.** Consider a finite binary tree T. Define L_T to be a set of all leaves of T. In case T consists of a single vertex ϵ, let $L_T = \{\epsilon\}$.

In particular, given a tree pair (D, R, t), we define:

1. the set L_D to be the set of all leaves of the tree D, and
2. the set L_R to be the set of all leaves of R.

As we view the tree D and the tree R of a tree pair (D, R, t) as finite subtrees of the infinite binary tree $\mathcal{T}$, we will view sets L_D and L_R as sets of nodes of $\mathcal{T}$. Note that this, in particular, implies that the sets L_D and L_R do not need to be disjoint.

With this terminology, and recalling the partial action which an $\alpha \in V$ induces on nodes of $\mathcal{T}$, we can conclude that α induces a bijection from L_D to L_R.

We will now introduce operations on trees. Suppose that (D, R, t) is a tree pair. Recall that both D and R are viewed as subtrees of $\mathcal{T}$. We consider the intersection $D \cap R$ to be the subgraph of both D and R such that $D \cap R$ consists of nodes and edges common to both trees D and R. Notice that $D \cap R$ is also a tree. This coincides with the usual definition of the intersection of graphs.

Lemma 2.3.6. *Suppose that (D, R, t) is a tree pair. A leaf of the tree $D \cap R$ is one of the following:*

1. *a leaf of the tree D and a proper ancestor of some leaves of the tree R,*
2. *a leaf of both the tree D and the tree R,*

3. *a leaf of the tree R and a proper ancestor of some leaves of the tree D.*

Proof. Suppose that λ is a leaf of the tree $D \cap R$. It means that λ is a node of both trees D and R. However, the nodes $\lambda 0$ and $\lambda 1$ are either not nodes the tree D or not nodes of the tree R, as if they were the nodes of both, they would belong to the tree $D \cap R$ and hence λ would not be one of its leafs.

Hence we have three possible cases:

1. The nodes $\lambda 0$ and $\lambda 1$ are not nodes of the tree D but they are nodes of the tree R. Hence, λ is a leaf of the tree D, and also a proper ancestor of some leaves of the tree R (as R is a finite tree).

2. The nodes $\lambda 0$ and $\lambda 1$ are neither nodes of the tree D nor the nodes of the tree R. In this case, λ is a leaf of both trees D and R.

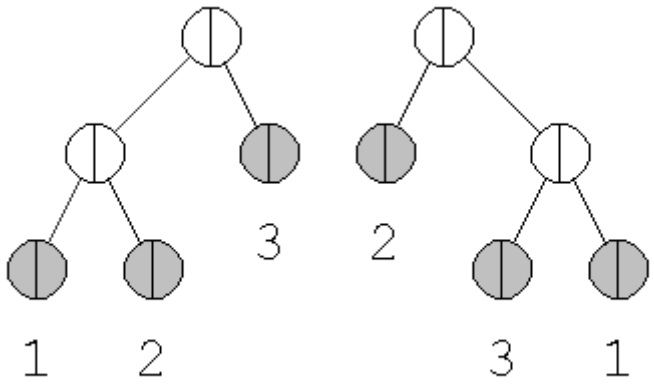
3. By the same argument as 1.

□

Let the difference of trees $D - R$ be defined as a (possibly empty) forest such that e is the edge of $D - R$ if and only if e is an edge of D and not an edge of R, and let v be a vertex of $D - R$ if and only if it is contained in one of the edges of $D - R$. Each tree of this forest is called a *component of* $D - R$. Observe that with this definition of $D - R$ some of the leaves of R might be vertices of $D - R$. We consider components of $D - R$ to be rooted at these vertices. For more detailed information, see Lemma 3.1.3. We define the difference of trees $R - D$ similarly.

Finally, let the union $D \cup R$ be a graph such that the vertex v belongs to $D \cup R$ if $v \in D$ or $v \in R$. Similarly, the edge e belongs to $D \cup R$ if $e \in D$ or $e \in R$. This is the usual definition for the union of graphs, so we expand it to any graphs.

Example 2.3.7. Consider the tree pair (D, R, t) from Example 2.3.2:



The set of vertices of the tree D is given by $\{\epsilon, 0, 00, 01, 1\}$. The set of edges of D is given by $\{\{\epsilon, 0\}, \{\epsilon, 1\}, \{0, 00\}, \{0, 01\}\}$. The set of vertices of the tree R is given by $\{\epsilon, 0, 1, 10, 11\}$. The set of edges of R is given by $\{\{\epsilon, 0\}, \{\epsilon, 1\}, \{1, 10\}, \{1, 11\}\}$.

Hence, the set of edges of the intersection $D \cap R$ is given by $\{\{\epsilon, 0\}, \{\epsilon, 1\}\}$, and the set of vertices of $D \cap R$ is given by $\{\epsilon, 0, 1\}$.

Consider the difference $D - R$. Its set of edges is the set of edges which belong to D but not to R, which is given by $\{\{0, 00\}, \{0, 01\}\}$. Its set of vertices is given by the set of all vertices present in its edges, namely $\{0, 00, 01\}$. Similarly, consider the difference $R - D$. Its set of edges is the set of edges which belong to R but not to D, which is given by $\{\{1, 10\}, \{1, 11\}\}$. Its set of vertices is given by the set of all vertices present in its edges, namely $\{1, 10, 11\}$.

Finally, consider the union $D \cup R$. Its set of vertices is given by

$$\{\epsilon, 0, 00, 01, 1, 10, 11\}$$

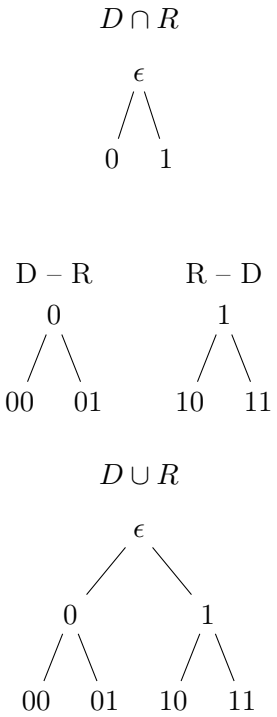
and its set of edges is given by

$$\{\{\epsilon, 0\}, \{\epsilon, 1\}, \{0, 00\}, \{0, 01\}, \{1, 10\}, \{1, 11\}\}$$

.

All of the graphs $D \cap R$, $D - R$ and $R - D$ and $D \cup R$ are given in

the following picture:



Finally, let us define:

Definition 2.3.8. Let T be a domain or range tree of a tree pair, and let v be a node of T.

Let us define vT_v as follows. The vertices of the tree vT_v are all vertices of T having prefix v, and the edges are all edges of T between the vertices with prefix v.

Let us now define a related tree T_v as follows. For each vertex vw ($w \in X^*$) of vT_v, the word w is a vertex of T_v. Also, for each edge between vw_1 and vw_2 ($w_1, w_2 \in X^*$) in vT_v, there is an edge between w_1 and w_2 in T_v.

Let $u \in X^*$. Let us also define the tree uT as follows. For each vertex w ($w \in X^*$) of T, the word uw is a vertex of uT. Also, for each edge between w_1 and w_2 ($w_1, w_2 \in X^*$) in T, there is an edge between uw_1 and uw_2 in uT.

Note that vT_v is a subtree of T, but it is rooted at v. The tree T_v bears the same 'shape' as the tree vT_v, but it is 'translated' to the root ϵ. The tree T_v is a subtree of $\mathcal{T}$, but not necessarily a subtree of T. The tree uT is a 'translation' of the tree T by the address u.

Equivalent Tree Pairs

We present an algorithm which illustrates how we can change a tree pair representing $\alpha \in V$ into another tree pair representing the same element α.

Algorithm 2.3.9 (Augmentation)**.** Let (D, R, t) be a tree pair representing element α of V. Let us define a new (D', R', t') as follows:

1. Pick a finite binary tree T, such that L_T is a barrier.
2. Pick a leaf λ of the tree D.
3. We define the new tree D' as the union $D \cup \lambda T$, and consider its root to be at ϵ.
4. We define the new tree R' as the union $R \cup (\lambda\alpha)T$, and consider its root to be at ϵ.
5. Let m' be number of leaves of the new tree D'. We label the leaves of D' with numbers from 1 to m' from left to right.
6. For each word $w \in L_T$, the expression λw is a new leaf of D'. Say λw has a new number l in D'. Then we let the leaf $(\lambda\alpha)w$ have the number l in R'.
7. For each $\lambda' \in L_D \backslash \{\lambda\}$, the leaf λ' is also a leaf of D'. Say it has a new number l in D'. Then we let the leaf $\lambda'\alpha$ have the number l in R'.
8. We obtain the m'-tuple t' by reading the numbers on the tree R' from left to right.

The initial tree pair (D, R, t) and the resultant tree pair(D', R', t') correspond to the same element α. This can be deduced by construction of (D', R', t').

One can view the resultant tree pair as an expansion of the initial tree pair, and we call the process an *augmentation*. If the set of leaves of the tree T is given by $L_T = \{0, 1\}$, then we call the algorithm a *simple augmentation*. A process which would reverse a simple augmentation is called a *simple reduction*, and we describe it in detail in Algorithm 3.2.28.

A simple augmentation from Algorithm 2.3.9 and a simple reduction from Algorithm 3.2.28 present basic tools for transitioning between tree

pairs representing same elements α of Thompson's group V. In Corollary 3.2.31 we prove that any tree pair for α can be transformed into any other tree pair for α using a finite sequence of simple reductions and augmentations.

Lemma 2.3.10. *If $\alpha, \beta \in V$ then $\alpha\beta \in V$.*

Proof. Let $\alpha \in V$ be represented by the triple $(\mathcal{D}_\alpha, \mathcal{R}_\alpha, t)$ and let $\beta \in V$ be represented by the triple $(\mathcal{D}_\beta, \mathcal{R}_\beta, \tau)$. If $\mathcal{R}_\alpha \neq \mathcal{D}_\beta$, then we can use (possibly repeatedly) Algorithm 2.3.9 on the tree pair (D_α, R_α, t) corresponding to the element α and the tree pair (D_β, R_β, τ) corresponding to the element β. We aim to obtain new tree pairs $(D'_\alpha, R'_\alpha, t')$ for α and $(D'_\beta, R'_\beta, \tau')$ for β in the following way. We perform the augmentation of (D_α, R_α, t) and the inverse image under the map α of each component of $D_\beta - R_\alpha$. Then the tree R'_α becomes the union $R_\alpha \cup (D_\beta - R_\alpha)$. Thus, it contains all edges and vertices from the tree D_β, as well as all edges and vertices from the tree R_α. Similarly, we perform the augmentation of (D_β, R_β, τ) and each component of $R_\alpha - D_\beta$. Then the tree D'_β becomes the union $D_\beta \cup (R_\alpha - D_\beta)$. Hence we get that $R'_\alpha = D'_\beta$.

Thus, suppose that the triple $(\mathcal{D}_\alpha, \mathcal{R}_\alpha, t)$ for α and the triple $(\mathcal{D}_\beta, \mathcal{R}_\beta, \tau)$ for β are already such that $\mathcal{R}_\alpha = \mathcal{D}_\beta$. Let m be the order of all four barriers involved. Let $\mathcal{D}_\alpha = \{u_1, u_2, \ldots, u_m\}$, $\mathcal{R}_\alpha = \{v_1, v_2, \ldots, v_m\} = \mathcal{D}_\beta$ and $\mathcal{R}_\beta = \{w_1, w_2, \ldots, w_m\}$, each in the lexicographic order. Hence, we have:

$$(u_{t_j})\alpha = v_j \qquad \text{and} \qquad (v_{\tau_i})\beta = w_i$$

for all $i, j \in \{1, 2, \ldots, m\}$. Therefore, we can combine these equations by substitution $j = \tau_i$ to obtain the identities:

$$(u_{t_{\tau_i}})\alpha\beta = (v_{\tau_i})\beta = w_i$$

for all $i \in \{1, 2, \ldots, m\}$. Hence, we define the m-tuple $T = (t_{\tau_1}, t_{\tau_2}, \ldots, t_{\tau_m})$, so that the composition $\alpha\beta$ is given by the triple (D_α, R_β, T), and $(u_{T_i})(\alpha\beta) = w_i$. This means that $\alpha\beta$ is a member of V. □

Corollary 2.3.11. *The set V is a group.*

Proof. We know that $V \subseteq \mathrm{Sym}(\mathfrak{C})$. The identity map $(\{\epsilon\}, \{\epsilon\}, (1))$ belongs to V, so V is non–empty. We know that if $\alpha \in V$ then $\alpha^{-1} \in V$

by Lemma 2.2.3. We also know that if $\alpha, \beta \in V$ then $\alpha\beta \in V$ by Lemma 2.3.10. Hence we conclude that V is a group. □

2.4 Family of Chameleon Groups

The family of Thompson's groups F, T and V was renamed as 'chameleon groups' by Brin in [10], with V known as the 'last chameleon'. We will now briefly introduce the other two chameleons.

We recall Definition 2.2.1 for a description of V. We define Thompson's group F as follows:

Definition 2.4.1. The group F is defined to be a subgroup of the group V consisting of all elements of the form $(\mathcal{D}, \mathcal{R}, t)$ such that $t = (1, 2, \ldots, m)$ in the natural order.

We also define Thompson's group T as follows:

Definition 2.4.2. The group T is defined to be a subgroup of the group V consisting of all elements of the form $(\mathcal{D}, \mathcal{R}, t)$ such that

$$t = (n, n+1, n+2, \ldots, m-1, m, 1, 2, \ldots, n-1)$$

for some $n \in \{1, 2, \ldots, m\}$.

As F and T are not central to this thesis, we omit proofs that they are groups.

Chapter 3

Dynamics via Combinatorics

In this chapter, we intend to familiarise the reader with the theory of dynamics occurring in the action of an element α of Thompson's group V on the Cantor space. This understanding will be essential for pursuing Chapter 4 on centralising α and Chapter 5 on exploring formation of free products in V.

We start by presenting Brin's [11] work on special kinds of tree pairs for elements of V, called *revealing pairs*. New terminology complementing and expanding Brin's is introduced alongside in Section 3.1 to describe behaviour occurring in non–revealing tree pairs. In particular, the new classification of *leaf chains*, called iterated augmentation chains in [11], allows us to detect and eliminate undesirable structures in a tree pair to transform it into a revealing pair. We achieve this in Section 3.2 by introducing and performing a series of algorithms on a new type of graph, called the *chains graph*, which encodes tree pair behaviour. These algorithms can also be used in different configurations for other purposes. In Section 3.3 we present work of Salazar–Díaz [28] on how distinct revealing pairs representing the same element α from V relate to each other. We describe it in terms of the newly developed terminology. Finally, in Section 3.4, we describe so–called *important points*, introduced by Bleak and Salazar–Díaz in [8], which are best detectable by using revealing pairs. Important points are of tremendous importance in Chapters 4 and 5.

3.1 Revealing Pairs

A *revealing pair* is a particular kind of tree pair representing an element α of Thompson's group V, introduced by Brin in [11], which *reveals* its

otherwise hidden dynamics. We expand Brin's terminology to describe all types of tree pairs.

In this section, we will develop basic terminology for leaves of trees which form a tree pair representing an element of V. Using this terminology, we will define a revealing pair of an element. We will finish by providing a full classification of leaves for a revealing pair and the types of sequences which these leaves form. We will provide an instance of a typical revealing tree pair in Example 3.1.13 and we will gradually explain its properties. All of this will help us better understand the dynamical patterns of points from $\mathfrak{C}$ under the action of an element of Thompson's group.

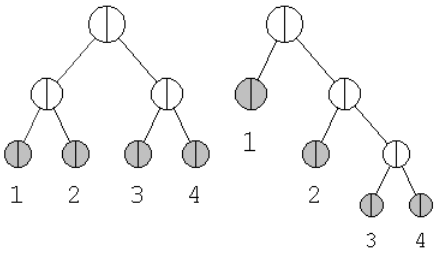
Leaves

Consider an element α in Thompson's group V. Recall from Definitions 2.2.1 and 2.3.1 how α can be represented by a tree pair (D, R, t), where D and R are finite binary trees with m leaves each, for some positive integer m, and t is an m-tuple representing an arrangement of the numbers from the set $\{1, \ldots, m\}$.

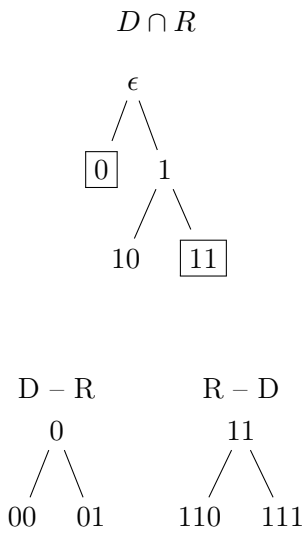
We can now proceed to start considering how we could discriminate between different types of leaves of a tree pair. Our naming conventions developed in this subsection will be essential for when we define a revealing tree pair in Definition 3.1.17.

Recall definitions of $D - R$ and $R - D$ from Section 2.3. Let us take the first step to begin the classification of leaves of a tree pair by analysing the following example:

Example 3.1.1 (Relative Position of Leaves in L_D and L_R)**.** Consider the tree pair (D, R, t) given by the picture below:



Consider the graphs $D \cap R$, $D - R$ and $R - D$:



Among the leaves of the domain tree D (on the left), the leaves 00 and 01, labelled with the numbers 1 and 2 respectively, are not leaves of the range tree R, but are leaves of a connected component of $D - R$. The leaf 10 of D, labelled with the number 4 in D, happens to be also a leaf of the tree R, labelled with the number 2 in R. Finally, the leaf 11 of D, labelled with the number 4, is not a leaf of R but it is a proper ancestor of some of the leaves of R.

Similarly, for leaves of the range tree R (on the right), the leaves 10 and 111, labelled with numbers 3 and 4 respectively, are not leaves of the domain tree D, but are leaves of a connected component of $R - D$. Finally, the leaf 0 of R, labelled with the number 1, is not a leaf of D but it is a proper ancestor of some of the leaves of D.

With the help of the example above, for a given tree pair (D, R, t) we will now show how to group leaves of D and leaves of R into categories. Let us start with the following definition:

Definition 3.1.2 (Neutral Leaf)**.** Consider a tree pair (D, R, t) representing an element α from V. If there is an address of a node of $\mathcal{T}$ which belongs to both set L_D and set L_R, then we call the leaf with this address a *neutral leaf* of the tree pair (D, R, t).

The leaves of D fall into three main categories, and the leaves of R also fall into three main categories, depending on their relative positions:

Lemma 3.1.3 (Relative Position of Leaves in L_D and L_R)**.** *Consider a tree pair (D, R, t) representing an element α of Thompson's group V. Each leaf of the tree D is one of the three categories:*

1. *a leaf of a component of $D - R$;*
2. *a neutral leaf;*
3. *a proper ancestor of a leaf of R and a root of a component of $R - D$.*

Similarly, each leaf of the tree R is one of the three categories:

1. *a leaf of a component of $R - D$;*
2. *a neutral leaf;*
3. *a proper ancestor of a leaf of D and a root of a component of $D - R$.*

Proof. Let us suppose that λ is a leaf of the tree D, then it is either a leaf of a component of $D - R$, which is Case 1), or a leaf of the intersection $D \cap R$. By Lemma 2.3.6, a leaf of the tree $D \cap R$ which is also a leaf of D can be either a leaf of both trees D and R, which is Case 2), or proper ancestor of some leaves of the tree R. In the latter case, it is also a root of a component of $R - D$, which is Case 3).

By symmetry, the same proof holds for a leaf of R. □

We will now define a special kind of a neutral leaf, namely a *periodic neutral leaf*:

Definition 3.1.4 (Periodic Neutral Leaf)**.** Consider a tree pair (D, R, t) representing an element α from V, and a neutral leaf λ of that tree pair. Then we call λ a *periodic neutral leaf* if there is a positive integer s such that:

1. for all integers i such that $0 \leq i \leq s$, the image $\lambda\alpha^i$ is a neutral leaf of that tree pair;
2. the leaf λ is mapped to itself by α^s, namely $\lambda = \lambda\alpha^s$.

Note that by this definition, all leaves $\lambda\alpha^i$ are *periodic neutral leaves.*

Note that if the number s from the definition above exists and is minimal, then λ, $\lambda\alpha$, $\lambda\alpha^2$, …, $\lambda\alpha^{s-1}$ are all distinct leaves of both the tree D and the tree R, and α permutes them cyclically.

Not all neutral leaves, however, satisfy the conditions for being periodic neutral leaves.

Definition 3.1.5 (Non–Periodic Neutral Leaf)**.** Consider a tree pair (D, R, t) representing an element α from V, and a neutral leaf λ of that tree pair. Then, we call λ a *non–periodic neutral leaf* if it is not a periodic neutral leaf.

Lemma 3.1.6. *Consider a tree pair (D, R, t) representing an element α from V. The non–periodic neutral leaves of this tree pair are precisely those neutral leaves that eventually get mapped outside the set of neutral leaves under some positive (and under some negative) power of α.*

Proof. We will prove the following equivalent statement. The periodic neutral leaves of the tree pair (D, R, t) are precisely those neutral leaves which do not get mapped outside the set of neutral leaves by any positive power of α or any negative power of α.

Suppose that λ is a periodic neutral leaf of (D, R, t). Then by Definition 3.1.4, there is a positive integer s such that $\lambda\alpha^i$ is a neutral leaf for all i such that $0 \leq i \leq s$, and $\lambda = \lambda\alpha^s$. Therefore, for any integer s', there exists an integer j such that $s' \equiv j \bmod s$ and $0 \leq j < s$. This means that $\lambda\alpha^{s'} = \lambda\alpha^j$, and so every image $\lambda\alpha^{s'}$ is a neutral leaf. Note that this statement is stronger than required.

Conversely, suppose that λ is a neutral leaf which does not get mapped outside the set of neutral leaves by any positive power of α. As the trees D and R have finitely many leaves, there are non–negative integers j and k with $j < k$ such that $\lambda\alpha^j = \lambda\alpha^k$. Take $s = k - j$. Then $\lambda\alpha^s = \lambda$ and $\lambda\alpha^i$ is a neutral leaf for all $0 \leq i \leq s$. Hence, λ is a neutral periodic leaf. The proof is the same if instead, we suppose that λ is a neutral leaf which does not get mapped outside the set of neutral leaves by any negative power of α. □

We will now start looking at the leaves from the leaf sets L_D and L_R which are not neutral leaves. The two definitions of non-neutral leaves, *repellers* and *attractors*, will be necessary for defining the central object of this section, namely a *revealing pair*:

Definition 3.1.7 (Repeller)**.** Consider an element α of V, represented by a tree pair (D, R, t), and a leaf λ of a component of $D - R$. Then we call λ a *repeller* if there is a positive integer s such that:

1. for all integers i such that $0 < i < s$, the image $\lambda\alpha^i$ is a neutral leaf;

2. the image $\lambda\alpha^s$ lies in $L_R \backslash L_D$ and is a proper ancestor of λ.

Definition 3.1.8 (Range of Repulsion)**.** Consider an element α of V, represented by a tree pair (D, R, t), and consider a repeller λ from the set L_D, and its forward orbit of leaves $\lambda\alpha^i$ for $0 \leq i \leq s$, such that: $\lambda\alpha^i$ is a neutral leaf for all i such that $0 < i < s$, and the image $\lambda\alpha^s$ lies in $L_R \backslash L_D$ and is a proper ancestor of λ. Then we call $\lambda\alpha^s$ a *range of repulsion.*

Notice that a range of repulsion is a root of a component of $D - R$. Also, a repeller is a leaf of a component of $D - R$. We have a collective name for all leaves of a component of $D - R$ which are not repellers:

Definition 3.1.9 (Source)**.** Consider an element α of V, and a tree pair (D, R, t) representing it. A *source* is a leaf of a component of $D - R$ which is not a repeller.

Similarly to Definition 3.1.7, we define:

Definition 3.1.10 (Attractor)**.** Consider an element α of V, represented by a tree pair (D, R, t), and a leaf λ of a component of $R - D$. Then we call λ an *attractor* if there is a negative integer r such that:

1. for all integers i such that $r < i < 0$, the image $\lambda\alpha^i$ is a neutral leaf;

2. the image $\lambda\alpha^r$ lies in $L_D \backslash L_R$ and is a proper ancestor of λ.

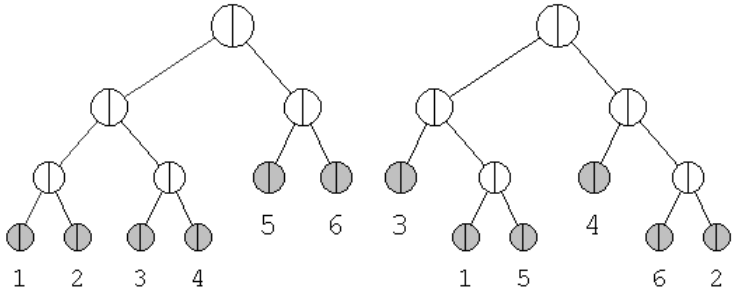
Definition 3.1.11 (Domain of Attraction)**.** Consider an element α of V, represented by a tree pair (D, R, t), and consider an attractor λ from the set L_R, and its backward orbit of leaves $\lambda\alpha^i$ for $r \leq i \leq 0$, such that: $\lambda\alpha^i$ is a neutral leaf for all i such that $r < i < 0$, and the image $\lambda\alpha^r$ lies in $L_D \backslash L_R$ and is a proper ancestor of λ. Then we call $\lambda\alpha^r$ a *domain of attraction.*

Notice that a domain of attraction is a root of a component of $R - D$. Also, an attractor is a leaf of a component of $R - D$. We have a collective name for all leaves of a connected component of $R - D$ which are not attractors:

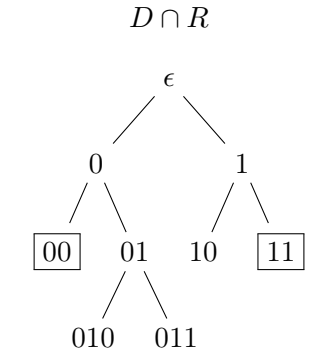
Definition 3.1.12 (Sink)**.** Consider an element α of V, and a tree pair (D, R, t) representing it. A *sink* is a leaf of a component of $R - D$ which is not an attractor.

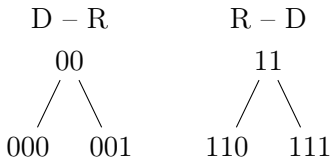
Below we present a worked example of a tree pair with the types of all its leaves identified:

Example 3.1.13 (Leaf Types)**.** Consider the tree pair (D, R, t) represented by the following picture:



Let us recall the definition of the difference of trees from Section 2.3 and hence identify components of $D - R$ and $R - D$.





It turns out that there is only one component of $D - R$, it is rooted at the node with address 00, and it is a single caret. Hence its leaves have addresses 000 and 001. Note that 00 is a leaf of the tree R.

Similarly, let us identify components of $R - D$. It turns out that there is also only one, it is rooted at the node with address 11, and it is a single caret. Hence its leaves have addresses 110 and 111. Note that 11 is a leaf

of the tree D. All the remaining leaves, namely the ones with addresses in the set $\{010, 011, 10\}$, are neutral leaves.

We will now identify prefix substitutions defined by this tree pair, in order to recognise the types of leaves involved:

$$\begin{array}{lll} (000)\alpha = 010 & (001)\alpha = 111 & (010)\alpha = 00 \\ (011)\alpha = 10 & (10)\alpha = 011 & (11)\alpha = 110 \end{array}$$

Let us systematically analyse the type of each of the leaves:

1. Leaves of $D - R$:

   (a) Leaf 000:

   $$(000)\alpha^2 = (010)\alpha = 00$$

   As 00 is a prefix of 000, *leaf* 000 *is a repeller.*

   (b) Leaf 001:

   $$(001)\alpha = 111$$

   As 111 is not a leaf of D and is not a prefix of 001, *leaf* 001 *is a source.*

2. Leaves of D which are proper ancestors of leaves of R:

   (a) Leaf 11:

   $$(11)\alpha = 110$$

   As 11 is an ancestor of 110, *leaf* 11 *is a domain of attraction.*

3. Neutral leaves:

   (a) Leaf 010:

   $$(010)\alpha = 00$$

   As 00 is not a leaf of D, *leaf* 010 *is a non–periodic neutral leaf.*

   (b) Leaf 011:

   $$(011)\alpha^2 = (10)\alpha = 011$$

   As 011 is mapped to itself by a positive power of α, *leaf* 011 *is a periodic neutral leaf.*

   (c) Leaf 10:

   $$(10)\alpha^2 = (011)\alpha = 10$$

   As 10 is mapped to itself by a positive power of α, *leaf* 10 *is a periodic neutral leaf.*

4. Leaves of $R - D$:

   (a) Leaf 110:

   $$(110)\alpha^{-1} = 11$$

   As 11 is an ancestor of 110, *leaf* 110 *is an attractor.*

(b) Leaf 111:

$$(111)\alpha^{-1} = 001$$

As 001 is not a leaf of R and is not a prefix of 111, *leaf* 111 *is a sink.*

5. Leaves of R which are proper ancestors of leaves of D:

(a) Leaf 00:

$$(00)\alpha^{-2} = (010)\alpha^{-1} = 000$$

As 00 is a prefix of 000, *leaf* 00 *is a range of repulsion.*

In the example above we observe each of the previously defined types of leaves.

It will be shown later in Lemma 3.1.20 that the previously defined leaves fully classify possible leaf types of a *revealing* pair.

Remark 3.1.14. For a general tree pair, consider Lemma 3.1.3: a leaf of a domain tree is either neutral (periodic or non–periodic), a leaf of a component of $D - R$ (repeller or source), or a root of a component of $R - D$. An example of the latter is a domain of attraction.

Similarly, a leaf of a range tree is either neutral (periodic or non–periodic), a leaf of a component of $R - D$ (attractor or sink), or a root of a component of $D - R$. An example of the latter is a range of repulsion.

Thus, there are two remaining types of leaves which might occur in a non–revealing tree pair (D, R, t) but have not been named:

1. A leaf of the tree R which is a root of a component of $D - R$ and which is not a range of repulsion.

2. A leaf of the tree D which is a root of a component of $R - D$ and which is not a domain of attraction.

We wanted to propose a name for each them, for the sake of completion and because we will be working with these types of leaves in this chapter:

Definition 3.1.15 (Range of Sourcing)**.** Consider an element α of V represented by a tree pair (D, R, t), and a leaf λ of the tree R which is a root of a component of $D - R$. If λ is not a range of repulsion, then we call it a *range of sourcing*.

Definition 3.1.16 (Domain of Sinking)**.** Consider an element α of V represented by a tree pair (D, R, t), and a leaf λ of the tree D which is a root of a component of $R - D$. If λ is not a domain of attraction, then we call it a *domain of sinking*.

The reason for this choice of names is, above all, consistency with the already existing terminology. In general, if a leaf λ is a range of sourcing, consider its backward orbit of leaves $\lambda\alpha^i$ for a negative integer r and $r \leq i \leq 0$ such that the image $\lambda\alpha^r$ lies in $L_D \backslash L_R$ and it is not a proper ancestor of λ. Then $\lambda\alpha^r$ can be either a source (hence the name *range of sourcing* for λ) or a domain of sinking. Similarly, if a leaf λ is a domain of sinking, consider its forward orbit of leaves $\lambda\alpha^i$ for a positive integer s and $0 \leq i \leq s$ such that the image $\lambda\alpha^s$ lies in $L_R \backslash L_D$ and it is not a proper ancestor of λ. Then $\lambda\alpha^s$ can be either a sink (thus the name *domain of sinking* for λ), or a range of sourcing.

Definition of a Revealing Pair

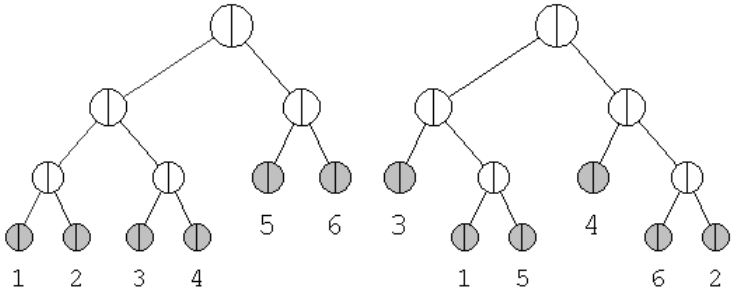
Now, we are prepared to define the key object of this section, known as a revealing pair, in terms of repellers and attractors:

Definition 3.1.17 (Revealing Pair for $\alpha \in V$)**.** Consider an element α of the Thompson group V, and consider a tree pair (D, R, t) representing α. Then we say that (D, R, t) is a *revealing pair for* α if:

1. every component of $D - R$ contains a repeller;

2. every component of $R - D$ contains an attractor.

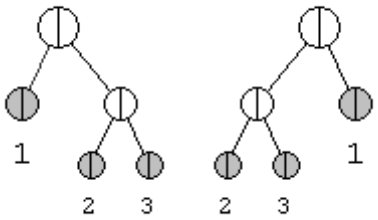
Example 3.1.18 (Revealing Pair)**.** Consider the tree pair (D, R, t) of Example 3.1.13 represented by the following picture:



Recall that there is only one connected component of $D - R$ and it has two leaves, 000 and 001. According to the analysis performed in Example 3.1.13, leaf 000 is a repeller. Similarly, recall that there is only one connected component of $R - D$ and it has two leaves, 110 and 111. According to the analysis performed in Example 3.1.13, leaf 110 is an attractor. Hence, the conditions for being a revealing pair are met by the tree pair (D, R, t).

For contrast, we will also give an example of a tree pair which fails to be revealing:

Example 3.1.19 (Non–Revealing Pair)**.** Consider the following tree pair (D, R, t) representing $\alpha \in V$:



There is precisely one connected component of $D - R$, and it is rooted at the address 1. It has precisely two leaves, with addresses 10 and 11. Now, $(10)\alpha = 00$, which is not a leaf of the tree D. It is also not an ancestor of the leaf 10, and so the leaf 10 is not a repeller. Similarly, $(11)\alpha = 01$, which is not a leaf of the tree D. It is also not an ancestor of the leaf 11, and so the leaf 11 is not a repeller. Hence, the connected

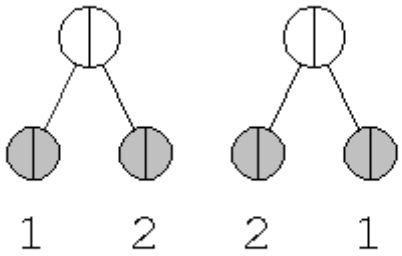
component of $D - R$ rooted at the address 1 has no repellers. Therefore, the tree pair (D, R, t) is not a revealing tree pair.

Notice that the tree pair gives the following prefix substitution for all $w \in \{0,1\}^\omega$:

$$(0w)\alpha = 1w \qquad (10w)\alpha = 00w \qquad (11w)\alpha = 01w$$

Consider these two rules that for all $w \in \{0,1\}^\omega$ we have $(10w)\alpha = 00w$ and $(11w)\alpha = 01w$. They could be replaced by a single rule that that for all $w \in \{0,1\}^\omega$ we have $(1w)\alpha = 0w$.

Hence the element α is in fact torsion of order two, and an example of its revealing tree pair is:



Note that both $D - R$ and $R - D$ are empty, so this tree pair indeed satisfies Definition 3.1.17.

Classification and Chains of Leaves

After defining a revealing pair, which is the object which motivates this section, we will take an opportunity to complete the classification of leaves of a revealing tree pair and describe types of chains which they form. It will prove to be helpful for understanding chains graphs in Section 3.2.

We will now complete the classification of revealing pair's leaves:

Lemma 3.1.20 (Types of Leaves of a Revealing Pair)**.** *If an element α of V is represented by a revealing tree pair (D, R, t), and if λ is an element of the union of the leaf sets $L_D \cup L_R$, then λ is of one of the following types:*

1. *a leaf of a component of $D - R$, which could be:*
   (a) *a repeller, or*
   (b) *a source;*
2. *a leaf of the tree D and at the same time a root of a component of $R - D$, which is a domain of attraction;*
3. *a leaf of the intersection of the tree D and the tree R, namely a neutral leaf, which could be:*
   (a) *periodic, or*
   (b) *non–periodic;*
4. *a leaf of the tree R and at the same time a root of a component of $D - R$, which is a range of repulsion;*
5. *a leaf of a component of $R - D$, which could be:*
   (a) *an attractor, or*
   (b) *a sink.*

Proof. First, recall Remark 3.1.14. Thus, we only need to show that every root of a component of $R - D$ is a domain of attraction and that every root of a component of $D - R$ is a range of repulsion. Given Definition 3.1.17 of a revealing pair, each component of $D - R$ contains a repeller, which is mapped to the root of this component by some positive power of α. Hence, this root is a range of repulsion. Observe that no other leaves of this component can be mapped to its root, and so they are sources. Similarly, each component of $R - D$ contains an attractor, which is mapped to the root of this component by some negative power of α. Hence, this root is a domain of attraction. Observe that no other leaves of this component can be mapped to its root, and so they are sinks. □

Note that the lemma above also proves the uniqueness of a repeller for a given a component of $D - R$, and the uniqueness of an attractor for a given component of $R - D$.

Now we know all the different types of leaves which can occur in a revealing tree pair. We also know the remaining types of leaves which may occur in a non–revealing tree pair: see Definitions 3.1.15 and 3.1.16. The next step which we are interested in is to determine what type of

patterns the leaves form. Consider the following algorithm for forming sequences of leaves, which we will call *chains*:

Algorithm 3.1.21. Consider a tree pair (D, R, t) representing an element α of Thompson's group V.

1. Pick a leaf $\lambda \in L_D \backslash L_R$ which has not been used in any of the previous chains. Let k be a positive integer such that the leaf $\lambda\alpha^k$ is a member of the set $L_R \backslash L_D$ and for all i such that $0 < i < k$ the leaf $\lambda\alpha^i$ is a neutral leaf. Then, create a chain $(\lambda\alpha^i)_{i=0}^k$.

2. Repeat the previous point until there are no more leaves left in $L_D \backslash L_R$ which have not been yet used for forming a chain.

3. Pick a leaf $\lambda \in L_D \cap L_R$ which has not been used in any of the previous chains. Let k be a non–negative integer such that for all i such that $0 \leq i \leq k$ the leaf $\lambda\alpha^i$ is a neutral leaf of the tree pair (D, R, t), and $i = k + 1$ is the smallest positive integer such that $\lambda = \lambda\alpha^i$. Then, create a chain $(\lambda\alpha^i)_{i=0}^k$.

4. Repeat the previous point until there are no more leaves left in $L_D \cap L_R$ which have not been yet used for forming a chain.

Note that we know from Lemma 3.1.6 that each non–periodic neutral leaf of (D, R, t) is a part of a chain formed in Step 1) of the algorithm above. Therefore, for the Step 3) we are only left with eligible periodic neutral leaves.

Definition 3.1.22 (Leaf Chain)**.** Consider a tree pair (D, R, t) representing and element α from V. Consider a chain $(\lambda\alpha^i)_{i=0}^k$ for some leaf $\lambda \in L_D$ and some non–negative integer k, formed by application of Algorithm 3.1.21 to the tree pair (D, R, t). Any such chain $(\lambda\alpha^i)_{i=0}^k$ is called a *leaf chain*.

Note that Brin [11] called such an object *iterated augmentation chain*, for the reason that its primary use was to perform an augmentation along the entire chain. We are still interested in this process, implicitly in Section 3.2 on transforming any tree pair into a revealing tree pair and explicitly in Section 3.3 on transforming one revealing pair into another revealing pair. However, we will use it also for deriving more information about a given tree pair, for instance about *important points* of α defined in Section 3.4.

More specifically, we can distinguish between the following two types of leaf chains:

Definition 3.1.23 (Non–Periodic Leaf Chain)**.** Consider a tree pair (D, R, t) representing and element α from V. Consider its leaf chain $(\lambda\alpha^i)_{i=0}^k$ created in Algorithm 3.1.21. If the chain was formed in Step 1), and hence $\lambda \in L_D \setminus L_R$ and $\lambda\alpha^k \in L_R \setminus L_D$, then such a chain is called a *non-periodic leaf chain*.

Definition 3.1.24 (Periodic Leaf Chain or P–Chain)**.** Consider a tree pair (D, R, t) representing an element α from V. Consider its leaf chain $(\lambda\alpha^i)_{i=0}^k$ created in Algorithm 3.1.21. If the chain was formed in Step 3), and hence $\lambda = \lambda\alpha^{k+1}$, then such a chain is called a *periodic leaf chain*, or simply a *P–chain*.

Note that a P–chain could consist of only one leaf.

In general, leaf chains create a partition for leaves of a given tree pair:

Lemma 3.1.25. *Consider a tree pair (D, R, t) representing an element α of Thompson's group V. An application of Algorithm 3.1.21 to the tree pair (D, R, t) partitions the leaves of the tree pair into disjoint leaf chains.*

Proof. We will first show that no two non–periodic leaf chains share any leaves. Then we will show that all leaves from the set $L_D \setminus L_R$, all leaves from the set $L_R \setminus L_D$, and all non–periodic neutral leaves of the tree pair (D, R, t) have been used to build non-periodic chain leaves. We will show that P–chains are formed from periodic neutral leaves exclusively and non–periodic leaf chains admit no periodic neutral leaves. Finally, we will show that no two P–chains share any leaves.

Let λ_1 and λ_2 be two distinct leaves from the set $L_D \setminus L_R$. Suppose that there are non–negative integers j_1, j_2 such that $\lambda_1\alpha^{j_1}$ and $\lambda_2\alpha^{j_2}$ are both leaves from the leaf chains which start at λ_1 and λ_2 respectively and that $\lambda_1\alpha^{j_1} = \lambda_2\alpha^{j_2}$. Without loss of generality, let $j_1 \leq j_2$. However, as α is a bijection, this implies that $\lambda_1 = \lambda_2\alpha^{j_2 - j_1}$. This implies that $j_1 = j_2$, as the leaf $\lambda_2\alpha^{j_2-j_1}$ needs to be a leaf from the set $L_D \setminus L_R$, just like λ_1 is. But then $\lambda_1 = \lambda_2$, which contradicts the initial assumption. Hence, any two non–periodic leaf chains have no leaves in common.

Note that $|L_D| = |L_R|$ which implies that $|L_D \setminus L_R| = |L_R \setminus L_D|$. As each non–periodic leaf chain finishes with a leaf from the set $L_R \setminus L_D$, we conclude that all leaves from this set have been used for forming

non–periodic leaf chains. Moreover, by Lemma 3.1.6, we must have used all non–periodic neutral leaves for building non–periodic neutral chains. Now consider a leaf from a P–chain. By Definition 3.1.4 each such leaf is a periodic neutral leaf. Hence a P–chain and a non–periodic leaf chain cannot have any leaves in common.

Finally, consider two P–chains $(\lambda_1\alpha^i)_{i=0}^{k_1}$ and $(\lambda_2\alpha^i)_{i=0}^{k_2}$, for some neutral leaves λ_1, λ_2 and some non–negative integers k_1, k_2. Suppose that the chain $(\lambda_2\alpha^i)_{i=0}^{k_2}$ was created after the chain $(\lambda_1\alpha^i)_{i=0}^{k_1}$. By the guidelines of Algorithm 3.1.21, we have $\lambda_2 \neq \lambda_1\alpha^i$ for all i such that $0 \leq i \leq k_1$. Now suppose that there exists j such that $\lambda_2\alpha^j = \lambda_1\alpha^i$ such that $0 \leq j \leq k_2$ for some i such that $0 \leq j \leq k_1$. But as α is a bijection and $\lambda_1 = \lambda_1\alpha^{k_1+1}$, we have $\lambda_2 = \lambda_1\alpha^{i-j}$. If $i-j \geq 0$, this immediately gives a contradiction. If $i-j \geq 0$, then there is an integer n such that $0 \leq i-j+n(k_1+1) \leq k_1$, and so $\lambda_1\alpha^{i-j} = (\lambda_1\alpha^{n(k_1+1)})\alpha^{i-j} = \lambda_1\alpha^{i-j+n(k_1+1)}$, which is also a contradiction.

Hence, all leaves of the tree pair (D, R, t) have been used to build leaf chains, and every two distinct chains are disjoint.

□

We will now define three distinct types of non–periodic leaf chains:

Definition 3.1.26 (Non–Periodic Chains)**.** Consider a tree pair (D, R, t) representing and element α from V. Consider a non–periodic leaf chain $(\lambda\alpha^i)_{i=0}^{k}$ for some leaf $\lambda \in L_D \backslash L_R$ and some positive integer k.

- If λ is a repeller of the tree pair (D, R, t), then such a chain is called an *R–chain*.
- If $\lambda\alpha^k$ is an attractor of the tree pair (D, R, t), then such a chain is called an *A–chain*.
- If λ is a source and $\lambda\alpha^k$ is a sink of the tree pair (D, R, t), then such a chain is called an *SS–chain*.

It will be shown later in Lemma 3.1.29 that the four types of chains from Definitions 3.1.24 and 3.1.26 fully classify possible leaf chains of a *revealing* tree pair. However, in a non–revealing pair, due to possible occurrence of a range of sourcing leaf (see Definition 3.1.15) or domain of sinking leaf (see Definition 3.1.16), we can encounter three more types of leaf chains:

Definition 3.1.27 (Chains of Non–Revealing Pair)**.** Consider a tree pair (D, R, t) representing an element α from V. Consider its non–periodic leaf chain $(\lambda\alpha^i)_{i=0}^k$ for some leaf $\lambda \in L_D \backslash L_R$ and some positive integer k.

- If λ is a source and $\lambda\alpha^k$ is a range of sourcing of the tree pair (D, R, t), then such a chain is called an *SRS–chain.*
- If λ is a domain of sinking and $\lambda\alpha^k$ is a sink of the tree pair (D, R, t), then such a chain is called a *DSS–chain.*
- If λ is a domain of sinking and $\lambda\alpha^k$ is a range of sourcing of the tree pair (D, R, t), then such a chain is called a *DSRS–chain.*

It is of special importance to acknowledge these type of chains, as in Section 3.2 on transforming a tree pair into a revealing tree pair they will be targeted for elimination via described algorithms.

We will now prepare for classifying all possible chains of a revealing tree pair with the following lemma:

Lemma 3.1.28. *Suppose that (D, R, t) is a revealing tree pair for an element α in V. Let $(\lambda\alpha^i)_{i=0}^k$ be a non–periodic leaf chain for some leaf $\lambda \in L_D \backslash L_R$ and some positive integer k. Then:*

- *The leaf λ is a repeller if and only if $\lambda\alpha^k$ is a range of repulsion.*
- *The leaf λ is a domain of attraction if and only if $\lambda\alpha^k$ is an attractor.*
- *The leaf λ is a source if and only if $\lambda\alpha^k$ is a sink.*

Proof. Let l be the number of components of $D - R$. As (D, R, t) is revealing, each of these components admits a repeller. Each repeller is the beginning of an R–chain, which ends in a range of repulsion leaf. As by Lemma 3.1.25 any two leaf chains are disjoint, the tree R admits at least l range of repulsion leaves. As l is the number of components of $D - R$, there cannot be more than l range of repulsion leaves. Hence, the tree R admits precisely l range of repulsion leaves and the tree D admits precisely l repellers. Thus, we can deduce that a non–periodic leaf chain starts with a repeller if and only if it finishes with a range of repulsion leaf.

Similarly, let l' be the number of components of $R - D$. As (D, R, t) is revealing, each of these components admits an attractor. Each attractor

is the end of an A–chain, which begins in a domain of attraction leaf. As by Lemma 3.1.25 any two leaf chains are disjoint, the tree D admits at least l' domain of attraction leaves. As l' is the number of components of $R - D$, there cannot be more than l' domain of attraction leaves. Hence, the tree D admits precisely l' domain of attraction leaves and the tree R admits precisely l' attractors. Thus, we can deduce that a non–periodic leaf chain starts with a domain of attraction leaf if and only if it finishes with an attractor.

Recall the classification of leaves of a revealing tree pair given in Lemma 3.1.20. By the pigeonhole principle, the leaf chain $(\lambda\alpha^i)_{i=0}^k$ starts with a source λ if and only if it ends with a sink $\lambda\alpha^k$. □

Note that as we considered all possibilities for the behaviour of a chain of leaves of a revealing tree pair, we now have a complete classification of chain types:

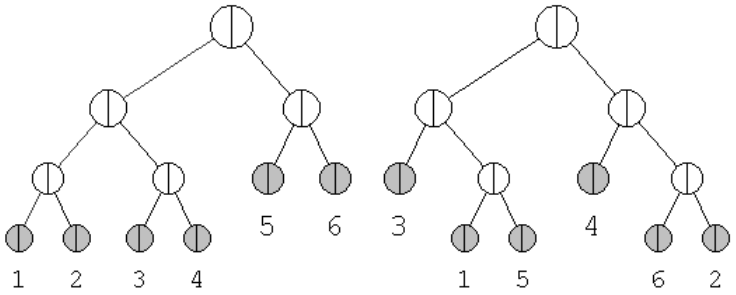
Corollary 3.1.29 (Types of Leaf Chains)**.** *Consider a revealing tree pair (D, R, t) representing an element α of V. Any leaf chain of (D, R, t) is one of the following:*

1. *P–chain;*
2. *R–chain;*
3. *A–chain;*
4. *SS–chain.*

Proof. By Lemma 3.1.20 and Lemma 3.1.28. □

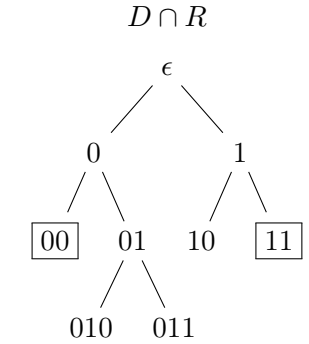
In order to allow a better intuitive understanding of different types of leaves and chains of a revealing tree pair, we provide the following example:

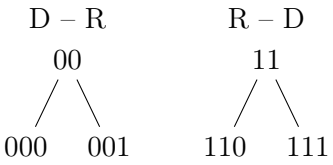
Example 3.1.30 (Leaf Chains)**.** Consider the tree pair (D, R, t) corresponding to an element α of V, represented by the following picture:



By Example 3.1.18, the tree pair (D, R, t) is revealing. Let us perform Algorithm 3.1.21 for finding its leaf chains.

First of all let us identify $D \cap R$, $D - R$ and $R - D$:





Now we can identify sets $L_D \cap L_R$, $L_D \backslash L_R$ and $L_R \backslash L_D$:

$$L_D \cap L_R = \{010, 011, 10\}$$
$$L_D \backslash L_R = \{000, 001, 11\}$$
$$L_R \backslash L_D = \{00, 110, 111\}$$

Recall that the prefix substitution rules for α are given by:

$$\begin{array}{lll} (000)\alpha = 010 & (001)\alpha = 111 & (010)\alpha = 00 \\ (011)\alpha = 10 & (10)\alpha = 011 & (11)\alpha = 110 \end{array}$$

Hence, we can identify the following leaf chains:

1. Non–periodic:
   (a) Start with the leaf 000: $000 \overset{\alpha}{\longmapsto} 010 \overset{\alpha}{\longmapsto} 00$.
   Here the leaf 010 is neutral and the leaf $00 \in L_R \setminus L_D$.
   (b) Start with the leaf 001: $001 \overset{\alpha}{\longmapsto} 111$.
   (c) Start with the leaf 11: $11 \overset{\alpha}{\longmapsto} 110$.
2. Periodic:
   (a) Start with the leaf 001: $001 \overset{\alpha}{\longleftrightarrow} 10$.

Let us now analyse each of the non–periodic leaf chains in order to decide on its further type:

1. (a) The chain $(000, 010, 00)$:
   The leaf 00 is a prefix of 000. Thus, 000 is a repeller, 00 is a range of repulsion and $(000, 010, 00)$ is an R–chain.
   (b) The chain $(001, 111)$:
   The leaf 001 is a leaf of a component of $D - R$ which is not a repeller (as 111 is not a prefix for it) so it is a source. The leaf 111 is a leaf of a component of $R - D$ which is not an attractor (as 001 is not a prefix for it) so it is a sink. Thus, $(001, 111)$ is an SS–chain.
   (c) The chain $(11, 110)$:
   The leaf 11 is a prefix of 110. Thus, 110 is an attractor, 11 is a domain of attraction, and $(11, 110)$ is an A–chain.

3.2 Algorithm for Obtaining a Revealing Pair

In this section, we will show a series of algorithms which transform any tree pair into a revealing tree pair representing the same element of

Thompson's group V. This will at the same time prove that revealing pairs for a given element of V always exist.

We will learn how to transform a tree pair into a *pre–chains* graph, and subsequently into a *chains graph*. The algorithms will rely on the use of chains graphs. After performing algorithms on the chains graph, we will be able to transform it back into a tree pair, which is revealing.

Pre–Chains Graph

Consider a tree pair (D, R, t) representing an element α of V. First of all, we are interested in translating it into a graph which contains all the data carried by the tree pair. The choice of working with a graph instead of a tree pair is motivated by a higher perceived clarity of patterns occurring in the type of graph which we construct.

Algorithm 3.2.1 (Pre–Chains Graph)**.** Consider a tree pair (D, R, t) representing an element α of Thompson's group V. We will construct a directed graph $G = (V_G, E_G)$ with a set of vertices V_G and a set of labelled edges E_G. We will allow multiple edges and loops.

1. Consider the intersection of the domain and range trees $D \cap R$. We define the set of vertices of the graph V_G to be the set $L_{D\cap R}$.

   Note that the set $V_G = L_{D\cap R}$ consists precisely of all neutral leaves of (D, R, t) and all roots of components of $D - R$ and $R - D$.

2. Each edge of the graph will be given by a triple, with the first and second entry being vertices of the graph, and the third entry being a label with metadata inherited from the tree pair. We will discriminate between four distinct categories of labels for edges of G, and we will represent that distinction by assigning a different label type to each category, namely DR, DD, RR and RD.

   Suppose that v is a vertex of the graph G.

   (a) Suppose further that v is a leaf of the domain tree D. Consider the image $(v)\alpha$ which is a leaf of the range tree R:

      i. *Label type DR.* If $(v)\alpha$ is a vertex of the graph G, then let there be an edge with label type DR from v to $(v)\alpha$, i.e. $(v, (v)\alpha, DR) \in E_G$.

ii. *Label type DD.* If $(v)\alpha$ is not a vertex of the graph G, then it must be a leaf of a component of $R - D$. Thus, there is a proper ancestor w of $(v)\alpha$ which is a vertex of the graph G. This implies that there is a word $s \in \{0,1\}^+$ such that $(v)\alpha = ws$. Then let there be an edge with a label type DD from v to w, namely $(v, w, DD(s)) \in E_G$.

(b) Suppose instead that v is not a leaf of the domain tree D. Then it must be a leaf of a range tree R, and a proper ancestor of some leaves of the domain tree D. Consider each word $s' \in \{0,1\}^+$ such that vs' is a leaf of the domain tree D, and consider its image $(vs')\alpha$.

i. *Label type RR.* If $(vs')\alpha$ is a vertex of the graph G, then there is an edge with a label type RR from v to $(vs')\alpha$, namely $(v, (vs')\alpha, (s')RR) \in E_G$.

ii. *Label type RD.* If $(vs')\alpha$ is not a vertex of the graph G, then it must be a leaf of a component of $R - D$. Thus, there is a proper ancestor w of $(vs')\alpha$ which is a vertex of the graph G. This implies that there is a word $s \in \{0,1\}^+$ such that $(vs')\alpha = ws$. Then let there be an edge with a label type RD from v to w, namely $(v, w, (s')RD(s)) \in E_G$.

3. Repeat Step 2) for each vertex v of the graph.

Definition 3.2.2 (Pre–Chains Graph)**.** Each graph which can be obtained from a tree pair representing an element of Thompson's group V following Algorithm 3.2.1 is called a *pre–chains graph.*

Lemma 3.2.3. *Algorithm 3.2.1 produces a unique graph G for a given tree pair (D, R, t).*

Proof. Suppose that λ is a leaf of the domain tree D.

1. Suppose further that $\lambda \in L_{D \cap R}$, which means that λ is a vertex of the graph G. Then by the inspection of the Algorithm 3.2.1 there is precisely one edge starting at λ, and there are two mutually exclusive possibilities for its label. If the unique edge starting at λ is (λ, l, DR) for some vertex $l \in V_G$, it means that $(\lambda)\alpha = l$ for this leaf l of the range tree R. If instead the unique edge starting at λ

is $(\lambda, l, DD(s))$, it means that $(\lambda)\alpha = ls$ for this vertex $l \in V_G$ and this word $s \in \{0,1\}^+$. The expression ls is also the address of the leaf of the range tree R, to which the leaf λ is mapped by α.

2. Suppose otherwise that $\lambda \notin L_{D\cap R}$, which means that λ is a leaf of a connected component of $D - R$ rooted at a leaf λ' of the range tree R such that $\lambda' \in L_{D\cap R}$, and there is a word $s' \in \{0,1\}^+$ such that $\lambda = \lambda' s'$. Then there are precisely two choices for the label of an edge of the graph G indicating where λ is mapped by α. If the edge is of the form $(\lambda', l, (s')RR)$, then $(\lambda)\alpha = (\lambda' s')\alpha = l$ for some leaf l of the range tree R. If instead the edge is of the form $(\lambda', l, (s')RD(s))$, then $(\lambda)\alpha = (\lambda' s')\alpha = ls$ for some $l \in V_G$ and a word $s \in \{0,1\}^+$. The expression ls is also the address of the leaf of the range tree R, to which the leaf λ is mapped by α.

□

Lemma 3.2.4. *For every pre–chains graph G, there is a unique tree pair (D, R, t) representing an element α of V, such that the application of Algorithm 3.2.1 to (D, R, t) results in the graph G. Moreover, we can identify (D, R, t) from the graph G.*

Proof. By definition of pre–chains graph, we may assume that there is a tree pair (D, R, t) for $\alpha \in V$ such that when we apply Algorithm 3.2.1 to it, it results in the graph G. We want to show that we can read all prefix substitutions for α precisely as given by the tree pair (D, R, t) from the pre–chains graph G.

We will show that for each edge of the pre–chains graph there is a leaf $\lambda \in L_D$ and a leaf $l \in L_R$ such that $\lambda\alpha = l$. We will also show that the collection of all these maps uniquely determines the tree pair (D, R, t).

Suppose that there is an edge between vertices v and w of the graph G. Then we have the following cases:

1. The edge is given by (v, w, DR). Then the vertex v is a leaf of D, the vertex w is a leaf of R and $(v)\alpha = w$.

2. The edge is given by $(v, w, DD(s))$ for some word $s \in \{0,1\}^+$. Then the vertex v is a leaf of D, the expression ws is a leaf of R and $(v)\alpha = ws$.

3. The edge is given by $(v, w, (s')RR)$ for some word $s' \in \{0,1\}^+$. Then the expression vs' is a leaf of D, the vertex w is a leaf of R and $(vs')\alpha = w$.

4. The edge is given by $(v, w, (s')RD(s))$ for some words $s, s' \in \{0,1\}^+$. Then the expression vs' is a leaf of D, the expression ws is a leaf of R and $(vs')\alpha = ws$.

Now notice that the set of vertices of a graph G is a complete set of leaves of an intersection of the domain tree and the range tree of the tree pair (D, R, t). This means that V_G is a barrier, as defined in Section 2.3. Recall Algorithm 3.2.1. By dichotomies presented between 2)*a*) and 2)*b*), 2)*a*)*i*) and 2)*a*)*ii*), and between 2)*b*)*i*) and 2)*b*)*ii*), each vertex $v \in V_G$ must be a beginning of an edge in G. We will show that for each such v, the edges of G uniquely determine the prefix substitution(s) for all points of $\mathfrak{C}$ underlying v.

Suppose that e is an edge with beginning v.

1. If the label type of e is DR or DD, then by Cases 1) and 2) above, this edge corresponds to a single prefix substitution for the prefix v, and we must have $v \in L_D$.

2. If the label type of e is RR or RD, then consider all edges which start at v. By 2)*b*) of Algorithm 3.2.1, these are given by:

$$\begin{aligned} e = e_1 &= (v, w_1, (s_1)R\ldots) \\ e_2 &= (v, w_2, (s_2)R\ldots) \\ &\vdots \\ e_x &= (v, w_x, (s_x)R\ldots), \end{aligned}$$

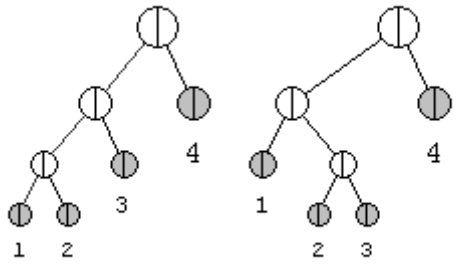
and the set $\{s_1, s_2, \ldots, s_x\}$ is a barrier. Therefore, we have

$$vs_1, vs_2, \ldots, vs_x \in L_D,$$

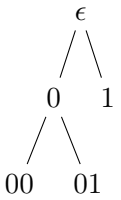
and we obtain a set of prefix substitutions for all of these leaves.

This determines prefix substitution(s) for all leaves of D with a given prefix v. As V_G is a barrier, we can recover prefix substitutions for all leaves of D, which proves our lemma. □

Example 3.2.5 (Pre–Chains Graph)**.** Consider the following tree pair (D, R, t) representing an element α of V:



First, we identify the intersection of trees D and R:



Hence, the set of vertices of the pre–chains graph G of the tree pair (D, R, t) is given by:

$$V_G = \{00, 01, 1\}$$

Now we proceed with the Algorithm 3.2.1 to identify edges of G:

1. Consider vertex 00. It is not a vertex of the domain tree. Its descendants which are vertices of the domain tree are 000 and 001. Let us consider each of them:

   (a) $(000)\alpha = 00$ and $00 \in V_G$ so $(00, 00, (0)RR) \in E_G$.

   (b) $(001)\alpha = 010$ and $010 \notin V_G$, but $01 \in V_G$ so

   $$(00, 01, (1)RD(0)) \in E_G.$$

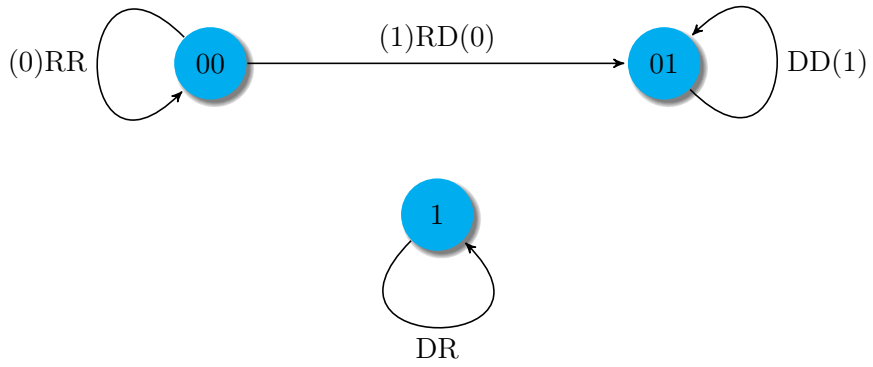
2. Consider vertex 01, which is a vertex of domain tree. $(01)\alpha = 011$ and $011 \notin V_G$ but $01 \in V_G$ so $(01, 01, DD(1)) \in E_G$.

3. Consider vertex 1, which is a vertex of domain tree. $(1)\alpha = 1$ and $1 \in V_G$ so $(1, 1, DR) \in E_G$.

Therefore, the set of edges of G is given by the set:

$$E_G = \{(00, 00, (0)RR), (00, 01, (1)RD(0)), (01, 01, DD(1)), (1, 1, DR)\}.$$

Thus we can summarise our results with the following picture:



The Pre–Chains Graph of the Tree Pair (D, R, t)

Properties of Pre–Chains Graphs

Let us expand the notation associated with pre–chains graphs. The objective is to be able to concisely and simultaneously refer to all types of edges.

Notation 3.2.6. Let us perform the following unification of the notation for edges of a chains graph:

1. An edge of a pre–chains graph of the form (u, w, DR) can be also denoted as $(u, w, ()DR())$.
2. An edge of a pre–chains graph of the form $(u, w, DD(s))$ can be also denoted as $(u, w, ()DD(s))$.
3. An edge of a pre–chains graph of the form $(u, w, (s)RR)$ can be also denoted as $(u, w, (s)RR())$.

Now we are in position to prove the following lemma:

Lemma 3.2.7. *Consider a tree pair (D, R, t) representing an element α of V, and its corresponding pre–chains graph G. Consider an edge*

$(u, w, (s')XY(s))$ of the graph G for some vertices $u, w \in V_G$, some letters $X, Y \in \{D, R\}$ and some words $s', s \in \{0,1\}^*$. Then both of the following hold:

1. *The word s' is empty if and only if $X = D$.*
2. *The word s is empty if and only if $Y = R$.*

Proof. By considering Algorithm 3.2.1 and Notation 3.2.6 we conclude that there are precisely four types of the edge labels. They are as following:

1. $(u, w, ()DR())$
2. $(u, w, ()DD(r))$
3. $(u, w, (r')RR())$
4. $(u, w, (r')RD(r))$

for some vertices $u, w \in V_G$ and some words $r', r \in \{0,1\}^+$. This proves the statement of the lemma. □

Before we present any transformations of a pre–chains graph, let us analyse some of its properties. They will follow naturally from properties of the associated tree pair.

Definition 3.2.8 (Neutral, Domain and Range Vertices)**.** Let (D, R, t) be a tree pair representing an element α of Thompson's group V. Let $G = (V_G, E_G)$ be the pre–chains graph of this tree pair.

1. If $v \in V_G$ is a neutral leaf of the tree pair (D, R, t), then we call it a *neutral vertex*.
2. If $v \in V_G$ is a leaf of the domain tree D, and a proper ancestor of a leaf of the range tree R, then we call it a *domain vertex*.
3. If $v \in V_G$ is a leaf of the range tree R, and a proper ancestor of a leaf of the domain tree D, then we call it a *range vertex*.

Lemma 3.2.9 (Neutral Vertex)**.** *Let (D, R, t) be a tree pair representing an element α of Thompson's group V. Let $G = (V_G, E_G)$ be the pre–chains graph of this tree pair. If $v \in V_G$ is a neutral vertex, then v has exactly one incoming edge with label type DR or RR (i.e., with the second letter R) and exactly one outgoing edge with label type DR or DD (i.e., with the first letter D).*

Proof. Suppose that v is a neutral leaf. This implies that v is a leaf of both the domain tree D and the range tree R. As v is a leaf of the range tree, by definition of the labelled edges, it cannot have an incoming edge with label DD or RD. Furthermore, by properties of a tree pair, there is a unique leaf u of the domain tree D such that $(u)\alpha = v$. If $u \in V_G$, then there is an edge from u to v with a label DR. If u is not a vertex of the graph G, there is a proper ancestor u' of u, such that $u' \in V_G$. In this case, there is an edge from u' to v with a label $(s')RR$, for the word $s' \in \{0,1\}^+$ such that $u's' = u$.

Similarly, as v is a leaf of a domain tree, by the definition of the label types of edges, it cannot have an outgoing edge with label RR or RD. As v is a leaf of the domain tree, then by properties of a tree pair there is a unique leaf u of the range tree R such that $(v)\alpha = u$. If $u \in V_G$, then there is an edge from v to u with a label DR. If u is not a vertex of the graph G, there is a proper ancestor u' of u, such that $u' \in V_G$. In this case, there is an edge from v to u' with a label $DD(s)$, for the word $s \in \{0,1\}^+$ such that $u's = u$.

□

Lemma 3.2.10 (Domain Vertex)**.** *Let (D, R, t) be a tree pair representing an element α of Thompson's group V. Let $G = (V_G, E_G)$ be the pre–chains graph of this tree pair. Suppose that $v \in V_G$ is a domain vertex. Then v has more than one incoming edge, each with label type DD or RD (i.e., with the second letter D) and exactly one outgoing edge with label type DR or DD (i.e., with the first letter D). Moreover, the exact number of incoming edges is equal to the number of leaves of the connected component of $R - D$ rooted at v.*

Proof. As v is not a leaf of the range tree R, it cannot have an incoming edge with label DR or RR. The vertex v is, in fact, a proper ancestor of more than one of the leaves of the tree R. Each of these leaves can be represented as vs, for some word $s \in \{0,1\}^+$. Then for each such vs, there is a unique vertex u of the domain tree D, such that $(u)\alpha = vs$. If u is a vertex of the graph G, then there is an edge $(u, v, DD(s)) \in E_G$. Suppose u is not a vertex of the graph G. Then there is a proper ancestor $u' \in V_G$ of the leaf u, such that $u's' = u$ for the word $s' \in \{0,1\}^+$. Then, $(u', v, (s')RD(s)) \in E_G$. As there is more than one proper descendant of v which is a leaf of the range tree R, there are more than one incoming edges, each with label DD or RD. As we quantify over s,

that exact number of incoming edges is equal to the number of leaves of the corresponding connected component of $R - D$ rooted at v.

For a proof of the case with a unique outgoing edge having the first letter of its label being D, the argument is the same as the second part of the proof of Lemma 3.2.9. □

Lemma 3.2.11 (Range Vertex)**.** *Let (D, R, t) be a tree pair representing an element α of Thompson's group V. Let $G = (V_G, E_G)$ be the pre–chains graph of this tree pair. Suppose $v \in V_G$ is a range vertex. Then v has exactly one incoming edge with label type DR or RR (i.e. with the second letter R) and more than one outgoing edge, each with label type RR or RD (i.e. with the first letter R). Moreover, the exact number of outgoing edges is equal to the number of leaves of the connected component of $D - R$ rooted at v.*

Proof. For a proof of the case with a unique incoming edge having the second letter of its label being R, the argument is the same as the first part of the proof of Lemma 3.2.9.

As v is not a leaf of the domain tree D, it cannot have an outgoing edge with label DD or DR. The vertex v is, in fact, a proper ancestor of more than one of the leaves of the tree D. Each of these leaves can be represented as vs', for some word $s' \in \{0,1\}^+$. Then for each such vs', there is a unique vertex u of the range tree R, such that $(vs')\alpha = u$. If u is a vertex of the graph G, then there is an edge $(v, u, (s')RR) \in E_G$. Suppose u is not a vertex of the graph G. Then there is a proper ancestor $u' \in V_G$ of the leaf u, such that $u's = u$ for the word $s \in \{0,1\}^+$. Then, $(v, u', (s')RD(s)) \in E_G$. As there is more than one proper descendant of v which is a leaf of the domain tree D, there are more than one outgoing edges, each with label RR or RD. As we quantify over s', that exact number of outgoing edges is equal to the number of leaves of the corresponding connected component of $D - R$ rooted at v. □

Remark 3.2.12 (Interpretation of Edge Labels). Let (D, R, t) be a tree pair representing an element α of the Thompson's group V. Let $G = (V_G, E_G)$ be the pre–chains graph of this tree pair.

Consider an edge $e \in E_G$. There are four possible label types for e. The label type reflects whether vertices which the edge is joining are leaves of the domain tree D or the range tree R.

First of all, if the first letter of the label type of e is a D, we know that the initial vertex is a leaf of the domain tree D. It might or might not be a vertex of the range tree as well. Otherwise, if the first letter of the label type of the edge e is an R, we know that the initial vertex is not a leaf of the domain tree, and hence by the construction of pre–chains graph it is a leaf of the range tree only.

Secondly, if the second letter of the label type of e is an R, we know that the end vertex is a leaf of the range tree R. It might or might not be a vertex of the domain tree as well. Otherwise, if the second letter of the label type of the edge e is a D, we know that the end vertex is not a leaf of the range tree, and hence by the construction of pre–chains graph it is a leaf of the domain tree only.

Chains Graph

We will now present how to further transform this pre–chains graph into an object called the *chains graph* which will be the main tool in the subsequent algorithms. This will involve replacing some neutral vertices and their incident edges with single edges, which labels record the original vertices. The chains graph is chosen over the pre–chains graph due to being easier to work with in our set up. On the other hand, the underlying reason for defining the pre–chains graph as an intermediate stage on the way to the chains graph is our perception of the greater clarity of the relationship between pre–chains graphs and tree pairs, especially through Lemmas 3.2.7, 3.2.9, 3.2.10 and 3.2.11. We will show later that all of these lemmas, and a few new ones, hold also for chains graphs.

Algorithm 3.2.13 (Chains Graph)**.** Let (D, R, t) be a tree pair representing an element α of V, and G be its pre–chains graph.

1. Recall that by Lemma 3.2.9 that every neutral vertex of G has a unique incoming edge. Consider a neutral vertex u_1 satisfying the additional condition that either:

   - its unique ingoing edge has label type RR, and then it is the form $(v, u_1, (s')RR)$ for a range vertex v and $s' \in \{0,1\}^+$;
     or:
   - its unique ingoing edge is of the form (v, u_1, DR) where $v \in V_G$ is a domain vertex. Note that we view this edge as $(v, u_1, ()DR)$.

Note that this composite condition happens precisely when the beginning of the unique ingoing edge of u_1 is not a neutral vertex.

Also, recall that by Lemma 3.2.9 that a neutral vertex of G has a unique outgoing edge. Thus, in particular, the vertex u_1 has a unique outgoing edge. For the neutral vertex u_i and its unique outgoing edge e_i, recursively define the following, starting with $i = 1$:

- If the end of the edge e_i is also a neutral vertex, call this end vertex u_{i+1}. Then the edge e_i is given by $e_i = (u_i, u_{i+1}, DR)$.
- If the end vertex of the edge e_i is not a neutral vertex, then call this vertex w, let k be the greatest integer such that u_k is defined, and terminate this recursive definition.

Then one of the following occurs:

- Either w is a range vertex and hence there is an edge

$$(u_k, w, DR) \in E_G,$$

- or w is a domain vertex and hence there is an edge

$$(u_k, w, DD(s)) \in E_G$$

with $s \in \{0,1\}^+$.

Now, perform the following procedures:

(a) Replace the edges

$$\begin{gathered}(v, u_1, (s')XR) \\ (u_1, u_2, DR) \\ \vdots \\ (u_{k-1}, u_k, DR) \\ (u_k, w, DY(s))\end{gathered}$$

for $X, Y \in \{D, R\}$ with a single edge

$$(v, w, (s')X(u_1, \ldots, u_k)Y(s)).$$

(b) Delete the vertices $u_1, \ldots, u_k$, each corresponding to a neutral vertex, from the set V_G.

2. Repeat Step 1) until there is no neutral vertex u_1 in G which satisfies the specified condition.

3. Pick a neutral vertex u_1 of the graph G. Note that we require that u_1 has not been previously deleted.

   For the neutral vertex u_i and its unique outgoing edge e_i, recursively define the following, starting with $i = 1$:

   - If the end of the edge e_i is not u_1, call this end vertex u_{i+1}. Then the edge e_i is given by $e_i = (u_i, u_{i+1}, DR)$.
   - If the end vertex of the edge e_i is u_1, let k be the greatest integer such that u_k has been defined so far, and terminate this recursive definition. Note that the edge e_i is given by $e_i = (u_k, u_1, DR)$.

   Then, perform the following procedures:

   (a) Replace the edges

   $$\begin{gathered}(u_1, u_2, DR)\\(u_2, u_3, DR)\\\vdots\\(u_k, u_1, DR)\end{gathered}$$

   with a single edge

   $$(u_1, u_1, D(u_2, u_3, \ldots, u_k)R).$$

   (b) Delete the vertices $u_2, u_3, \ldots, u_k$, each corresponding to a neutral vertex, from the set V_G.

4. Repeat the previous step until there is no unused neutral vertex in G.

Definition 3.2.14 (Chains Graph)**.** Each graph which can be obtained from a pre–chains graph of a tree pair representing an element of Thompson's group V by following Algorithm 3.2.13 is called a *chains graph*.

Remark 3.2.15. Notice that in Steps 1) and 2) of the Algorithm 3.2.13 by Lemma 3.1.6 we got rid of all vertices of G representing non–periodic neutral leaves of the tree pair. Hence in Step 3) we have only these vertices to consider, which correspond to periodic neutral leaves of the tree pair. Hence for each remaining neutral vertex u_1 we can always find described sequence of leaves.

Lemma 3.2.16. *Consider an input of a tree pair (D, R, t) representing element α of V into Algorithm 3.2.1, and subsequently applying Algorithm 3.2.13 to the produced pre–chains graph. Let the resulting chains graph be called G. Then, for each tree pair (D, R, t) there exists a unique chains graph G, up to labels of vertices representing periodic orbits. Moreover, the tree pair (D, R, t) can be identified from the graph G.*

Proof. We already know by Lemmas 3.2.3 and 3.2.4 that a given pre–chains graph corresponds uniquely to a tree pair which it was constructed from. Hence, it is sufficient to show that a given chains graph corresponds uniquely to the pre–chains graph which it was constructed from. Consider both types of edges which are produced by Algorithm 3.2.13.

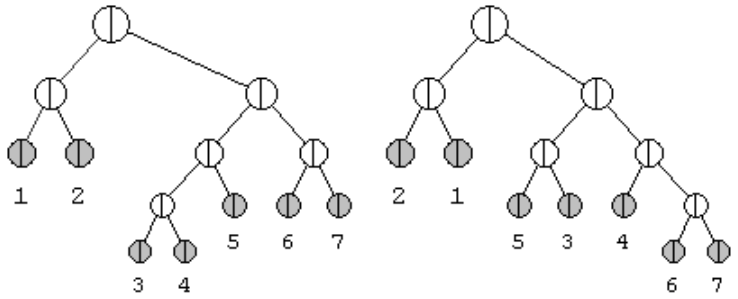
Each of the produced edges is of the form

$$(v, w, (s')X(u_1, \ldots, u_k)Y(s))$$

for some (not necessarily distinct) vertices $v, w \in V_G$, some $s', s \in \{0, 1\}^*$, some $X, Y \in \{D, R\}$, a non–negative integer k and neutral vertices $u_1, \ldots, u_k$ of the pre–chains graph obtained from the tree pair (D, R, t). Such an edge represents the following prefix substitutions: $(vs')\alpha = u_1, (u_1)\alpha = u_2, \ldots, (u_k)\alpha = ws$. It is the same as for the original, now replaced, edges of the pre–chains graph.

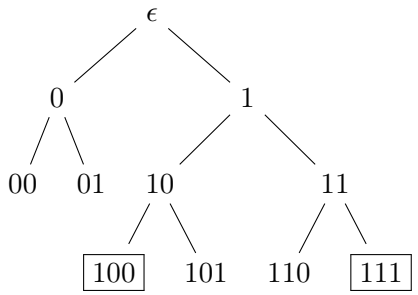
Hence, for each tree pair (D, R, t) there is a unique chains graphs G derived from it, and the tree pair (D, R, t) can be identified from the graph G. □

Example 3.2.17 (Chains Graph)**.** Consider the following tree pair (D, R, t) representing an element α of V:



First, we will construct the pre–chains graph for this tree pair, and subsequently we will transform it into the chains graph.

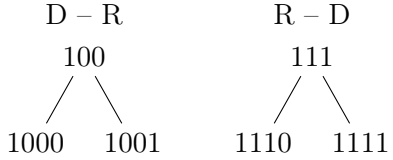
The intersection of the trees D and R is given by:



Hence, the set of vertices of the pre–chains graph G of the tree pair (D, R, t) is given by:

$$V_G = \{00, 01, 100, 101, 110, 111\}.$$

Let us also identify the forests $D - R$ and $R - D$:



Now we proceed with Algorithm 3.2.1 to identify the edges of the pre–chains graph G:

1. Consider the vertex 00, which is a leaf of the domain tree. We have $(00)\alpha = 01$ and $01 \in V_G$, so

$$(00, 01, DR) \in E_G.$$

2. Consider the vertex 01, which is a leaf of the domain tree. We have $(01)\alpha = 00$ and $00 \in V_G$, so

$$(01, 00, DR) \in E_G.$$

3. Consider the vertex 100. It is not a leaf of the domain tree. Its descendants which are vertices of the domain tree are 1000 and 1001.

   (a) We have $(1000)\alpha = 101$ and $101 \in V_G$, so

$$(100, 101, (0)RR) \in E_G.$$

   (b) We have $(1001)\alpha = 110$ and $110 \in V_G$, so

$$(100, 110, (1)RR) \in E_G.$$

4. Consider the vertex 101, which is a leaf of the domain tree. We have $(101)\alpha = 100$ and $100 \in V_G$, so

$$(101, 100, DR) \in E_G.$$

5. Consider the vertex 110, which is a vertex of the domain tree. We have $(110)\alpha = 1110$ and $1110 \notin V_G$, but $111 \in V_G$. Thus,

$$(110, 111, DD(0)) \in E_G.$$

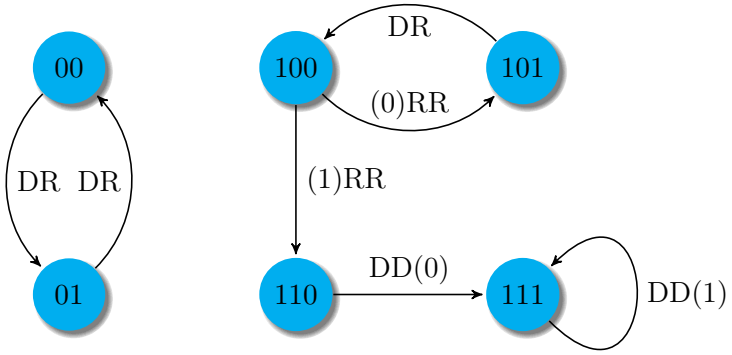
6. Consider the vertex 111, which is a vertex of the domain tree. We have $(111)\alpha = 1111$ and $1111 \notin V_G$, but $111 \in V_G$. Thus,

$$(111, 111, DD(1)) \in E_G$$

This process results in the following set of edges:

$$E_G = \{(00, 01, DR), (01, 00, DR), (100, 101, (0)RR), (100, 110, (1)RR),$$
$$(101, 100, DR), (110, 111, DD(0)), (111, 111, DD(1))\}$$

Therefore, the pre–chains graph G is given by the following picture:



The Pre–Chains Graph of the Tree Pair (D, R, t)

By inspection we identify the set of all neutral leaves of the tree pair (D, R, t) to be: $\{00, 01, 101, 110\}$. Notice that these are precisely the vertices of the pre–chains graph which have exactly one ingoing and exactly one outgoing edge.

Finally, we proceed with Algorithm 3.2.13 to transform G into the chains graph of the tree pair (D, R, t):

1. Let us start with considering criteria for Step 1) of the algorithm:

   (a) We pick vertex 00 for consideration. Its unique incoming edge is $(01, 00, DR)$ and 01 is a neutral leaf of the tree pair (D, R, t). Thus vertex 00 does not meet the required criterion for Step 1).

   (b) We pick vertex 01 for consideration. Its unique incoming edge is $(00, 01, DR)$ and 00 is a neutral leaf of the tree pair (D, R, t). Thus vertex 01 does not meet the required criterion for Step 1).

(c) We pick vertex 101 for consideration. Its unique incoming edge is $(100, 101, (0)RR)$. Hence 101 meets the criterion for Step 1).

i. The unique outgoing edge of 101 is $(101, 100, DR)$. The vertex 100 does not correspond to a neutral leaf of (D, R, t).

ii. We replace the edges $(100, 101, (0)RR)$ and $(101, 100, DR)$ with a single edge $(100, 100, (0)R(101)R)$.

iii. We delete the vertex 101 from the set V_G.

(d) We pick vertex 110 for consideration. Its unique incoming edge is $(100, 110, (1)RR)$. Hence 110 meets the criterion for Step 1).

i. The unique outgoing edge of 110 is $(101, 111, DD(0))$.

ii. We replace the edges $(100, 110, (1)RR)$ and $(101, 111, DD(0))$ with a single edge $(100, 111, (1)R(110)D(0))$.

iii. We delete the vertex 110 from the set V_G.

2. We considered all possible neutral vertices to meet criterion for Step 1) and we made all required changes. Hence we proceed to Step 3)

3. We pick a neutral vertex which has not met the criterion for Step 1), say 00.

(a) The unique incoming edge of 00 is $(01, 00, DR)$. The unique incoming edge of vertex 01 is $(00, 01, DR)$.

(b) We replace the edges $(01, 00, DR)$ and $(00, 01, DR)$ with a single edge $(00, 00, D(01)R)$.

(c) We delete the vertex 01 from the set V_G.

4. There are no remaining neutral vertices which have not been deleted or used. Hence we terminate the algorithm.

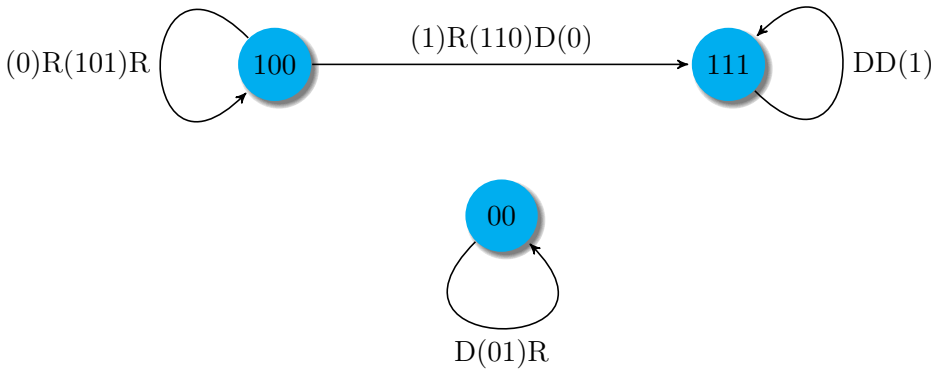
Note that in the second step the vertex 00 has been considered and stayed in V_G, while the vertex 01 has been deleted before there was a need to consider it. Hence, the current graph G is the chains graph of the tree pair (D, R, t). Its set of vertices is given by:

$$V_G = \{00, 100, 111\}$$

Also, its set of edges is given by:

$$E_G = \{(00, 00, D(01)R), (100, 100, (0)R(101)R),$$
$$(100, 111, (1)R(110)D(0)), (111, 111, DD(1))\}$$

Hence we have the following picture of the chains graph corresponding to the given tree pair (D, R, t):



The Chains Graph of the Tree Pair (D, R, t)

Properties of the Chains Graph

We want to emphasise that Algorithm 3.2.13 could be instead written in terms of a single local operation:

Remark 3.2.18 (Concatenation of Edges). We notice that Algorithm 3.2.13 has the effect of multiple application of concatenation of two edges joined by a neutral vertex of a graph G which initially represents the pre–chains graph. More precisely, we would obtain the same effect as the algorithm, if every time we see a vertex v corresponding to a neutral leaf of the initial tree pair and distinct edges $(u, v, (s^-)XR)$ and $(v, w, DY(s^+))$ for some vertices u, w of the pre–chains graph, some words $s^-, s^+ \in \{0, 1\}^*$ and $X, Y \in \{D, R\}$, we concatenate them into a single edge $(u, w, (s^-)X(v)Y(s^+))$ and cancel vertex v from the graph. The concatenation applies in the same way to already concatenated edges, which means that every time we see a vertex v corresponding to a neutral

leaf of the initial tree pair and distinct edges $(u, v, (s^-)X(u_1, \ldots, u_k)R)$ and $(v, w, D(u'_1, \ldots, u'_{k'})Y(s^+))$ for some vertices u, w of the intermediate graph between pre-chains graph and chains graph, some neutral leaves $u_1, \ldots, u_k, u'_1, \ldots, u'_{k'}$ of the initial tree pair, some words $s^-, s^+ \in \{0,1\}^*$, $X, Y \in \{D, R\}$ and some non-negative integers k, k', we concatenate them into a single edge

$$(u, w, (s^-)X(u_1, \ldots, u_k, v, u'_1, \ldots, u'_{k'})Y(s^+))$$

and cancel the vertex v. Notice in particular how two successive edges with label types XR and DY respectively turn into a single edge with label type XY, which acknowledges the cancelled intermediate vertex in the bracket between the letters. This phenomenon occurs implicitly also in Algorithms 3.2.34, 3.2.44, 3.2.51, 3.2.57 and 3.2.64, which are presented later on in this section.

The definition below is a restatement of notation given initially for pre–chains graphs:

Definition 3.2.19 (Domain, Range and Neutral Vertices)**.** Consider a tree pair (D, R, t) representing an element α of V, and its chains graph G. Let v be a vertex of the graph G.

1. We call v *neutral vertex* if it is a neutral leaf of the tree pair (D, R, t).
2. We call v a *domain vertex* if it is a leaf of the domain tree D but is not a leaf of the range tree R.
3. We call v a *range vertex* if it is a leaf of the range tree R but is not a leaf of the domain tree D.

The lemma below is a restatement of Lemmas 3.2.7, 3.2.9, 3.2.10 and 3.2.11 for chains graphs:

Lemma 3.2.20. *Consider a tree pair (D, R, t) representing an element α of V, and its corresponding chains graph G.*

1. *Consider an edge $(u, w, (s')X(u_1, \ldots, u_k)Y(s))$ of the graph G for some vertices $u, w \in V_G$, some letters $X, Y \in \{D, R\}$, some neutral leaves $u_1, \ldots, u_k$ and some words $s', s \in \{0,1\}^*$. Then both of the following hold:*
   (a) *The word s' is empty if and only if $X = D$.*

(b) *The word s is empty if and only if $Y = R$.*

2. *If $v \in V_G$ is a neutral vertex, then v has exactly one incoming edge with label type DR or RR and exactly one outgoing edge with label type DR or DD.*

3. *If $v \in V_G$ is a domain vertex, then v has more than one incoming edge, each with label type DD or RD and exactly one outgoing edge with label type DR or DD.*

4. *If $v \in V_G$ is a range vertex, then v has exactly one incoming edge with label type DR or RR and more than one outgoing edge, each with label type RR or RD.*

Proof. Given Remark 3.2.18 we notice that all the properties of label types edges of chains graph used in proofs of Lemmas 3.2.7, 3.2.9, 3.2.10 and 3.2.11 are still valid. □

Lemma 3.2.21. *Consider a tree pair (D, R, t) representing an element α of V, its chains graph G, and a neutral vertex v of the graph G. Then, there is a unique edge of G which both starts and finish at v. This edge represents full periodic orbit of neutral leaves (or a leaf) of the tree pair (D, R, t).*

Proof. In the pre–chains graph, every neutral leaf of the tree pair (D, R, t) was represented by a vertex. In Remark 3.2.15 about Algorithm 3.2.13 we observe that each vertex corresponding to a non–periodic neutral leaf has been deleted. Also, each vertex representing a periodic neutral leaf which has not been deleted is the one chosen to be the start and end of a new edge. This edge is unique and of the form $(v, v, D(u_1, \ldots, u_k)R)$ for some non–negative integer k and some neutral leaves $u_1, \ldots, u_k$ of the tree pair (D, R, t). This indicates that $(v)\alpha = u_1$, $(u_1)\alpha = u_2$, …, $(u_k)\alpha = v$. This is by definition a finite periodic orbit of neutral leaves. Note that if $k = 0$, then v is fixed by α, and so on a finite orbit of size one.

□

We present the following corollary as a useful summary:

Corollary 3.2.22 (Types of Vertices)**.** *At the end of Algorithm 3.2.13 applied to the pre–chains graph of a tree pair (D, R, t) to produce the*

chains graph G, we distinguish between precisely three types of vertices of our chains graph G:

1. *A neutral vertex with a single edge starting and ending at it.*

2. *A domain vertex corresponding to a leaf of the tree $D \cap R$ which is a leaf of the domain tree D but not a leaf of the range tree R.*

3. *A range vertex corresponding to a leaf of the tree $D \cap R$ which is a leaf of the range tree R but not a leaf of the domain tree D.*

Proof. Consider the fact that each vertex v of the chains graph G is a leaf of the tree $D \cap R$. Furthermore, there three options: v can be a leaf of domain tree D which is not a leaf of range tree R (a domain vertex), a leaf of range tree R which is not a leaf of domain tree D (a range vertex), or neutral leaf of (D, R, t) (a neutral vertex). Now by Lemma 3.2.21 we further conclude that if v is a neutral vertex of the graph G, then there is a single edge starting and ending at v. This proves the corollary. □

Definition 3.2.23 (Chain Edge)**.** Consider a tree pair (D, R, t) representing an element α of V, and its chains graph G. An edge e of the graph G is called a *chain edge.*

The reason for giving a collective name of a 'chain edge' for each of the edges of chains graphs is the following correspondence:

Lemma 3.2.24. *Consider a tree pair (D, R, t) representing an element α of V, and its chains graph G. Each edge of the graph G corresponds to a leaf chain of the tree pair (D, R, t).*

Proof. Recall Definition 3.1.22. Consider Algorithm 3.1.21 for finding leaf chains applied to a tree pair (D, R, t) and rules for formation of chain edges.

We observe that there is a non–periodic leaf chain $(\lambda\alpha^i)_{i=0}^k$ for some leaf $\lambda \in L_D \backslash L_R$ and a positive integer k if and only if there is an edge $(\lambda^-, \lambda^+, (s^-)X(\lambda_1, \ldots, \lambda_l)Y(s^+)) \in V_G$ for a non–neutral vertex λ^- of the graph G and a word $s^- \in \{0,1\}^*$ such that $\lambda^- s^- = \lambda$, for $l = k-1$, for neutral leaves λ_i of the tree pair (D, R, t) for each positive integer i such that $0 < i < k$ such that $\lambda_i = \lambda\alpha^i$ and for vertex λ^+ of the graph G and a word $s^+ \in \{0,1\}^*$ such that $\lambda^+ s^+ = \lambda\alpha^k$.

We also deduce that there is a P–chain $(\lambda\alpha^i)_{i=0}^k$ for some leaf $\lambda \in L_D \cap L_R$ and a positive integer k if and only if there is an edge

$$(\lambda_0, \lambda_0, D(\lambda_1, \ldots, \lambda_l)R) \in V_G$$

for $l = k$, for vertex $\lambda_0 \in V_G$ such that $\lambda_0 = \lambda\alpha^j$ for some non–negative integer j such that $0 \leq j \leq k$ and neutral leaves λ_i of the tree pair (D, R, t) such that $\lambda_i = \lambda\alpha^{j+i}$ for each positive integer i such that $0 < i \leq l$.

□

We draw particular attention to the following case:

Corollary 3.2.25. *Consider a chains graph G corresponding to the tree pair (D, R, t) representing an element α of Thompson's group V. Suppose that there is an edge e of the label type DR starting and finishing at the same vertex. Then, the edge e corresponds to a P–chain of leaves.*

Proof. By Lemma 3.2.21, Lemma 3.2.24 and Definition 3.1.24. □

Note that the chains graph G may have other edges of DR type which do not start and finish at the same vertex.

Remark 3.2.26. Let $(\lambda\alpha^i)_{i=0}^k$ be a non–periodic leaf chain of a tree pair (D, R, t) for some positive integer k. Let e be the edge of chains graph G of this tree pair, corresponding to the given leaf chain.

Notice that the initial leaf λ of the leaf chain is always a leaf of domain tree D and never a leaf of range tree R. However, λ might or might not be a vertex of the chains graph G, depending on whether λ is a leaf of the tree $D \cap R$ or not. In case when it is a leaf of the tree $D \cap R$, it is a vertex of G, and the first letter of label type of e is D. In case when λ is not a leaf of the tree $D \cap R$, it is not a vertex of G, and the first letter of label type of e is R.

Similarly, notice that the last leaf $\lambda\alpha^k$ of the leaf chain is always a leaf of range tree R and never a leaf of domain tree D. However, $\lambda\alpha^k$ might or might not be a vertex of the chains graph G, depending on whether $\lambda\alpha^k$ is a leaf of the tree $D \cap R$ or not. In case when it is a leaf of the tree $D \cap R$, it is a vertex of G, and the last letter of label type of e is R. In case when λ_k is not a leaf of the tree $D \cap R$, it is not a vertex of G, and the last letter of label type of e is D.

Under the correspondence of Lemma 3.2.24, we can prove the following:

Lemma 3.2.27. *Consider a tree pair (D, R, t) representing $\alpha \in V$ and its chains graph G. An edge of G with label type RD corresponds to an SS–chain.*

Proof. Recall Definition 3.1.26 of an SS–chain. Suppose that

$$e = (v, w, (s^-)R(u_1, \ldots, u_l)D(s^+))$$

is an edge of G for some $v, w \in V_G$, some non–negative integer l, some words $s^-, s^+ \in \{0,1\}^+$ and some neutral leaves $u_1, \ldots, u_l$ of the tree pair (D, R, t). By Lemma 3.2.20, as v has an outgoing edge with the label type starting with R, the vertex v is a range vertex. Similarly, by Lemma 3.2.20, as w has an incoming edge with the label type ending with D, the vertex w is a domain vertex. As vs^- is not a leaf of the range tree and ws^+ is not a leaf of the domain tree, the edge e corresponds to a non–periodic leaf chain $(vs^-, u_1, \ldots, u_l, ws^+)$. We want to show that vs^- must be a source and that ws^+ must be a sink. By Lemma 3.1.14, as v is a range vertex, vs^- is a leaf of a component of $D - R$, and hence it can be either a repeller or a source. But ws^+, the last leaf in the leaf chain starting at vs^-, is not a proper ancestor of vs^-, hence vs^- must be a source. Similarly, by Lemma 3.1.14, as w is a domain vertex, ws^+ is a leaf of a component of $R - D$, and hence it can be either an attractor or a sink. But vs^-, the first leaf in the leaf chain ending at ws^+, is not a proper ancestor of ws^+, hence ws^+ must be a sink.

As vs^- is a source and ws^+ is a sink, by Definition 3.1.26 the leaf chain $(vs^-, u_1, \ldots, u_l, ws^+)$ corresponding to the edge

$$(v, w, (s^-)R(u_1, \ldots, u_l)D(s^+))$$

is an SS–chain. □

We will describe further correspondences between leaf chain types and chain edges at points when they will be specifically used.

The Algorithm Part I – Maximal Tree Reduction

Consider a tree pair (D, R, t) representing an element α of Thompson's group V. We will now present the first of three parts of the algorithm,

which will transform (D, R, t) into a revealing tree pair which still represents the same $\alpha \in V$. The aim of this part is to find a tree pair representing α which domain and range trees have the smallest possible number of leaves. It is not an essential part of the process, but it often results in less data to analyse, as well as fewer leaves in the domain and range trees of the final revealing tree pair. This part of the algorithm will be introducing changes directly to the tree pair (D, R, t), as opposed to the chains graph, in the form of iterated simple reduction.

Algorithm 3.2.28. Consider a tree pair (D, R, t) representing an element α of Thompson's group V. Suppose that there are words w_1, w_2 in $\{0, 1\}^*$ such that all of the following hold:

- The addresses $w_1 0$ and $w_1 1$ are leaves of the tree D.
- The addresses $w_2 0$ and $w_2 1$ are leaves of the tree R.
- We have $(w_1 0)\alpha = w_2 0$ and $(w_1 1)\alpha = w_2 1$.

Then we will perform a *simple reduction*, given by:

1. Consider the tree D. Delete the vertices $w_1 0$ and $w_1 1$ of D and delete the edges $(w_1, w_1 0)$ and $(w_1, w_1 1)$ of D. Let the resulting tree be called D'. Note that the node w_1 is a leaf of D'.
2. Consider the tree R. Delete the vertices $w_2 0$ and $w_2 1$ of R and delete the edges $(w_2, w_2 0)$ and $(w_2, w_2 1)$ of R. Let the resulting tree be called R'. Note that the node w_2 is a leaf of R'.
3. We need to change t accordingly, so the new tree pair represents the same element α. Let us suppose that the leaves $w_1 0$ and $w_1 1$ were labelled with numbers k and $k + 1$ respectively, for some k such that $1 \leq k \leq m - 1$. (They need to have consecutive labels, as they are consecutive leaves in the tree D.) Now, perform the following changes to the m-tuple t:
   (a) Suppose that $i \in \{1, 2, \ldots, m\}$ and $1 \leq t_i \leq k$. In this case, we leave t_i as it is.
   (b) Suppose that for some $i \in \{1, 2, \ldots, m\}$, we have $t_i = k + 1$. In this case, we delete this coordinate t_i from the m-tuple t, creating an $(m - 1)$–tuple.

(c) Now, consider all $i \in \{1, 2, \ldots, m\}$ such that $k+2 \leq t_i \leq m$. Simultaneously for all i, replace each occurrence of t_i with $t_i - 1$.

(d) We call the resulting $(m-1)$-tuple t'.

4. We now replace (D, R, t) by our new tree pair (D', R', t'). The number m is now redefined to be such that the new t is an m-tuple. This means that m has decreased by 1.

After each simple reduction, we test the new tree pair for existence of such words w_1 and w_2. We terminate the algorithm when no such words exist.

Definition 3.2.29 (Reduced Tree Pair)**.** Consider a tree pair (D, R, t) representing an element α of Thompson's group V. If (D, R, t) remains unchanged by Algorithm 3.2.28, then we call it a *reduced tree pair*.

Lemma 3.2.30. *Given input of any tree pair (D, R, t) representing an element α of V, Algorithm 3.2.28 terminates with output of the unique reduced tree pair representing α.*

Proof. With each iteration of a simple reduction we decrease the positive integer m associated with the tree pair by 1. The number m cannot become non–positive by performing the described reductions of the tree pair. This is because the minimal number of leaves of D (and R) needed to apply the procedure is $m = 2$. Hence the algorithm must terminate. We know that the reduction can be applied when we can find words $w_1, w_2 \in \{0,1\}^*$ satisfying the criteria that:

1. $w_1 0$ and $w_1 1$ are leaves of the domain tree D,
2. $w_2 0$ and $w_2 1$ are leaves of the range tree R,
3. for all $w' \in \{0,1\}^\omega$, we have $(w_1 0 w')\alpha = w_2 0 w'$ and $(w_1 1 w')\alpha = w_2 1 w'$.

After application of a reduction, the tree pair (D, R, t) changes to (D', R', t'), but we are interested in showing that it still represents the same element α. We know that now for all $w' \in \{0,1\}^\omega$, we have $(w_1 w')\alpha = w_2 w'$, which is equivalent to the third condition above. We also observe that for each other leaf λ of the domain tree D which is also a leaf of the tree D', the effect of α is the same by construction of t'. This is because if λ

was a leaf with a number l which is less than the number of the leaf $w_1 0$ in the tree D, then λ still has number l in the tree D' and the leaf $(\lambda)\alpha$ has number l in both trees R and R'. If in contrast λ was a leaf with a number l which is bigger than the number of the leaf $w_1 1$ in the tree D, then λ has number $l-1$ in the tree D' and the leaf $(\lambda)\alpha$ which had number l in the tree R, has number $l-1$ in tree R'. This means that λ is still mapped to $(\lambda)\alpha$ by the new tree pair. As all of these define where all the points of $\mathfrak{C}$ are mapped to under the new tree pair, and all of them are being mapped in exactly same way as by the initial tree pair (D, R, t) and final tree pair (D', R', t'), both tree pairs must represent the same element α of V.

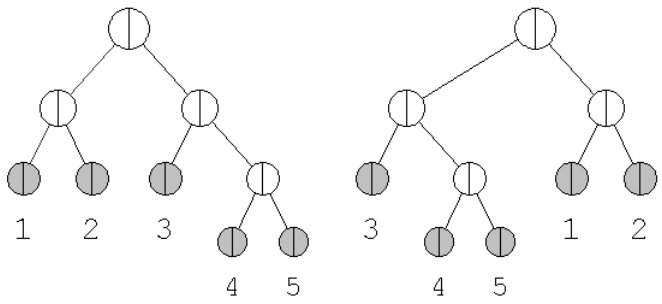
We will now show uniqueness of the reduced tree pair corresponding to any tree pair representing fixed element α of V. For that purpose, we will paraphrase and expand the proof from Cannon, Floyd and Parry [14] of existence of a unique reduced tree pair for any element of Thompson's group F. Let us now assume that (D, R, t) is a reduced tree pair representing $\alpha \in V$. The proof is based on the fact that if the action of α on all elements of $\mathfrak{C}$ underlying a given address could be extended to a partial map on that given address, then this address cannot be a non–leaf node of the tree D. More formally, the conditions for Algorithm 3.2.28 to stop with the tree pair (D, R, t) are equivalent to the following statement: if an address $u \in \{0,1\}^*$ is such that we have some address $v \in \{0,1\}^*$ such that for all $w \in \{0,1\}^\omega$ we have $(uw)\alpha = vw$, then no descendant of u is a leaf of the tree D. This means that u is either a leaf of D, or it is a proper descendant of a leaf of D. Now, as α can be interpreted as a set of prefix substitutions, this statement provides a unique choice of the set of prefixes pairs $\{(u_1, v_1), \ldots, (u_k, v_k)\}$ for some positive integer k, such that for all integers i such that $1 \leq i \leq k$ we have $(u_i)\alpha = v_i$ and $\{u_1, \ldots, u_k\} = L_D$ for some tree D as well as $\{v_1, \ldots, v_k\} = L_R$ for some tree R. This gives a unique reduced tree pair (D, R, t) corresponding to a given element α of V. □

Corollary 3.2.31. *Let α be an element of V and let (D, R, t) and (D', R', t') be tree pairs both representing the element α. Then there is a sequence of simple augmentations and simple reductions which transforms the tree pair (D, R, t) into the tree pair (D', R', t').*

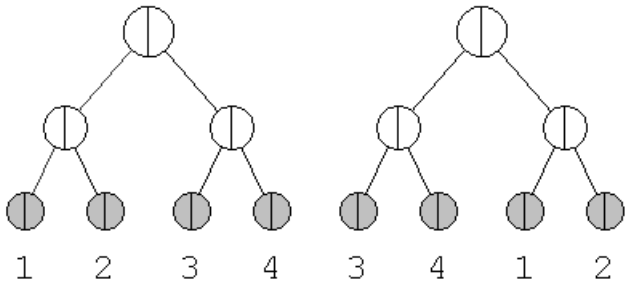
Proof. Recall that a simple augmentation defined in Algorithm 2.3.9 and a simple reduction defined in Algorithm 3.2.28 are inverse operations.

By Lemma 3.2.30 both (D, R, t) and (D', R', t') admit a finite sequence of simple reductions to the same reduced tree pair for α. This proves our claim. □

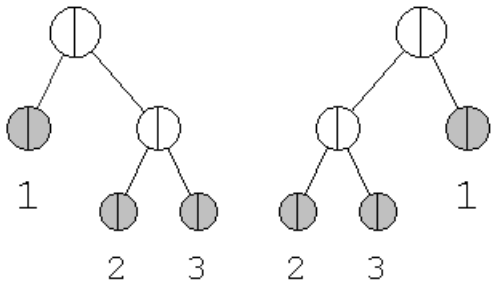
Example 3.2.32 (Reducing Tree Pair)**.** Consider the following tree pair (D, R, t):



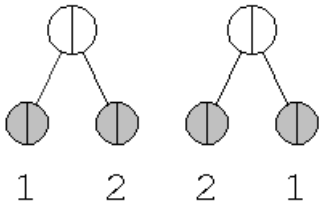
At the moment the procedure of Algorithm 3.2.28 can be applied to two pairs of addresses (w_1, w_2): $(0, 1)$ and $(11, 01)$. After the application to $(11, 01)$, we are left with the tree:



After the subsequent application of the algorithm to $(0, 1)$, we are left with the tree pair:



Now we recognise that a new pair of vertices (w_1, w_2) satisfying the criteria for algorithm's application has been uncovered, namely $(1, 0)$. After applying the algorithm to this last one, we have the following tree pair:



At this point, the only candidate for a pair of nodes which could satisfy the criteria for application of the algorithm's procedure is (,), namely a pair of empty nodes. However, this would require all points in $\mathfrak{C}$ to be fixed by α, which is not the case. Hence, we computed the reduced tree pair for the initial tree pair.

The Algorithm Part II – Detecting Torsion

Consider a tree pair (D, R, t) representing an element α of Thompson's group V, and its corresponding chains graph G. As noted earlier, we will not assume that (D, R, t) is reduced, but typically one applies Algorithm 3.2.28 first for efficiency. We will now present an algorithm which will modify G and result in another chains graph corresponding to another tree pair, which still represents the same element α of V, and which can be identified from the resulting chains graph. The new chains graph will admit no edges with label type DR which do not represent full periodic

orbits of its neutral leaves. We will also show that applying this algorithm we will also detect all P–chains which will occur in the final revealing tree pair. Therefore, after application of this algorithm we can easily decide whether the element α is torsion.

Recall that by Corollary 3.2.25 any edge with label type DR and the same start and end vertex represents a full periodic orbit of neutral leaves corresponding to a P–chain. However:

Lemma 3.2.33. *For a tree pair (D, R, t) representing $\alpha \in V$ and its chains graph G, an edge with label type DR with distinct start and end vertex corresponds to a $DSRS$–chain.*

Proof. Recall Definition 3.1.27 for a $DSRS$–chain. Suppose that

$$(v, w, D(u_1, \ldots, u_l)R)$$

is an edge of G for some $v, w \in V_G$ such that $v \neq w$, some non–negative integer l and some neutral leaves $u_1, \ldots, u_l$ of the tree pair (D, R, t). We know by Corollary 3.2.22 that v and w cannot be neutral vertices of G. Hence, v is a domain vertex and w is a range vertex. Recall that by Lemma 3.2.24 that this edge corresponds to a non–periodic leaf chain $(v, u_1, \ldots, u_l, w)$. We want to show that v is a domain of sinking leaf and that w is a range of sourcing leaf. As v is a domain vertex, it is a root of a component of $R - D$, and hence it can be either domain of attraction or domain of sinking leaf. But w, the last leaf in the leaf chain starting at v, is not a proper descendant of v, hence v must be a domain of sinking. Similarly, as w is a range vertex, it is a root of a component of $D - R$, and hence it can be either range of repulsion or range of sourcing leaf. But v, the first leaf in the leaf chain ending at w, is not a proper descendant of w, hence w must be a range of sourcing.

As v is a domain of sinking and w is a range of sourcing, by Definition 3.1.27 the leaf chain $(v, u_1, \ldots, u_l, w)$ corresponding to the edge

$$(v, w, D(u_1, \ldots, u_l)R)$$

is a $DSRS$–chain. □

As we know that $DSRS$–chains cannot occur in revealing tree pairs. The Algorithm 3.2.34 is aimed at eliminating these edges.

Algorithm 3.2.34. Consider a tree pair (D, R, t) representing an element α of Thompson's group V, and its corresponding chains graph G. Consider an edge of the graph G with a label type DR.

1. If the start and end vertex of the edge are the same, we leave it as it is.

2. Suppose that the start vertex v of the edge and the end vertex w of the edge are distinct. Suppose further that the edge joining them is given by $(v, w, D(u_1, \ldots, u_k)R)$ for some non–negative integer k and for some neutral leaves $u_1, \ldots, u_k$. Then by Corollary 3.2.22 the vertex v is a leaf of the domain tree D but not of the range tree R, and the vertex w is a leaf of the range tree R but not of the domain tree D.

   Recall from Definition 2.3.8 that we can denote the connected component of $R - D$ rooted at v to be vR_v, and recall that R_v is then the tree vR_v 'translated' to the root ϵ. Similarly, recall from Definition 2.3.8 that we can denote the connected component of $D - R$ rooted at w to be wD_w, and recall that D_w is then the tree wD_w 'translated' to the root ϵ. Define the intersection of the trees R_v and D_w by $T = R_v \cap D_w$.

   We will now list all leaves of T. Notice that as $T = R_v \cap D_w$, each leaf of T is either a leaf of the tree R_v or of the tree D_w, and possibly both. Let $p_1, p_2, \ldots, p_{l_1}$ for a non–negative integer l_1 be addresses of all leaves of T which are in the same time leaves of R_v, but not leaves of D_w. Let $q_1, q_2, \ldots, q_{l_2}$ for a non–negative integer l_2 be addresses of all leaves of T which are in the same time leaves of D_w, but not leaves of R_v. Finally, let $r_1, r_2, \ldots, r_{l_3}$ for a non–negative integer l_3 be all leaves of T which are in the same time leaves of R_v and leaves of D_w. Note that the collection of all these leaves forms a set of all leaves of the tree T.

   Perform the following action:

   (a) Introduce new vertices to the graph G, namely wp_i for all $i \in \{1, 2, \ldots, l_1\}$, and vq_j for all $j \in \{1, 2, \ldots, l_2\}$.

Recall that p_i is a leaf of the tree R_v, for all $i \in \{1, 2, \ldots, l_1\}$. Hence

for each such p_i there is a unique edge

$$(u', v, (s')X(u'_1, \ldots, u'_{k_1})D(p_i))$$

for some $u' \in V_G$, some non–negative integer k_1, some word $s' \in \{0,1\}^*$, some neutral leaves $u'_1, \ldots, u'_{k_1}$ of the tree pair (D, R, t), and some $X \in \{D, R\}$.

Perform the following action:

(b) For each p_i, replace the edge

$$(u', v, (s')X(u'_1, \ldots, u'_{k_1})D(p_i))$$

with the edge

$$(u', wp_i, (s')X(u'_1, \ldots, u'_{k_1}, vp_i, u_1p_i, \ldots, u_kp_i)R).$$

Recall by definition of p_i that wp_i is a proper ancestor of some leaves of the tree D. Now, recall Definition 2.3.8 and consider the tree D_{wp_i}. For each leaf s of the tree D_{wp_i}, we have an edge

$$(w, w', (p_is)R(w_1, \ldots, w_{l_w})Y(s_w))$$

in the modified graph G, for some $w' \in V_G$, some non–negative integer l_w, some addresses $w_1, \ldots, w_{l_w}$, $Y \in \{D, R\}$ and some word $s_w \in \{0,1\}^*$.

Perform the following action:

(c) For each p_i and each leaf s of D_{wp_i}, replace the edge

$$(w, w', (p_is)R(w_1, \ldots, w_{l_w})Y(s_w))$$

with the edge

$$(wp_i, w', (s)R(w_1, \ldots, w_{l_w})Y(s_w)).$$

Recall that q_j is a leaf of the tree D_w, for all $j \in \{1, 2, \ldots, l_2\}$. Hence for each such q_j there is a unique edge

$$(w, u', (q_j)R(u''_1, \ldots, u''_{k_2})Y(s''))$$

for some $u' \in V_G$, some non–negative integer k_2, some word $s'' \in \{0,1\}^*$, some neutral leaves $u''_1, \ldots, u''_{k_2}$ of the tree pair (D, R, t), and some $Y \in \{D, R\}$.

Perform the following action:

(d) For each q_j, replace the edge

$$(w, u', (q_j)R(u''_1, \ldots, u''_{k_2})Y(s''))$$

with the edge

$$(vq_j, u', D(u_1q_j, \ldots, u_kq_j, wq_j, u''_1, \ldots, u''_{k_2})Y(s'')).$$

Recall by definition of q_j that vq_j is a proper ancestor of some leaves of the tree R. Now, recall Definition 2.3.8 and consider the tree R_{vq_j}. For each leaf s of the tree R_{vq_j}, we have an edge

$$(v', v, (s_v)X(v_1, \ldots, v_{l_v})D(q_j s))$$

in the modified graph G, for some $v' \in V_G$, some non–negative integer l_v, some addresses $v_1, \ldots, v_{l_v}$, $X \in \{D, R\}$ and some word $s_v \in \{0,1\}^*$.

Perform the following action:

(e) For each q_j, replace the edge

$$(v', v, (s_v)X(v_1, \ldots, v_{l_v})D(q_j s))$$

with the edge

$$(v', vq_j, (s_v)X(v_1, \ldots, v_{l_v})D(s))$$

Finally, recall that r_h is a leaf of both trees R_v and D_w, for all $h \in \{1, 2, \ldots, l_3\}$. Hence for each such r_h there is a unique edge

$$(u', v, (s')X(u'_1, \ldots, u'_{k_1})D(r_h))$$

for some $u' \in V_G$, some non–negative integer k_1, some word $s' \in \{0,1\}^*$, some neutral leaves $u'_1, \ldots, u'_{k_1}$ of the tree pair (D, R, t),

and some $X \in \{D, R\}$. For each such r_h there is also a unique edge

$$(w, u'', (r_h)R(u''_1, \ldots, u''_{k_2})Y(s''))$$

for some $u'' \in V_G$, some non-negative integer k_2, some word $s'' \in \{0,1\}^*$, some neutral leaves $u''_1, \ldots, u''_{k_2}$ of the tree pair (D, R, t), and some $Y \in \{D, R\}$. Note that these two edges do not need to be distinct.

Perform the following actions:

(f) If the edges

$$(u', v, (s')X(u'_1, \ldots, u'_{k_1})D(r_h))$$

and

$$(w, u'', (r_h)R(u''_1, \ldots, u''_{k_2})Y(s''))$$

are distinct, then replace these two edges with a single edge

$$(u', u'', (s')X(u'_1, \ldots, u'_{k_1}, vr_h, u_1r_h, \ldots, \ldots, u_kr_h, wr_h, u''_1, \ldots, u''_{k_2})Y(s'')).$$

(g) If the edges

$$(u', v, (s')X(u'_1, \ldots, u'_{k_1})D(r_h))$$

and

$$(w, u'', (r_h)R(u''_1, \ldots, u''_{k_2})Y(s''))$$

are the same edge, then introduce the vertex vr_h to the graph and consider both of them be given by

$$(w, v, (r_h)R(u'_1, \ldots, u'_{k_1})D(r_h))$$

In this case, replace this edge with a single edge

$$(vr_h, vr_h, D(u_1r_h, \ldots, u_kr_h, wr_h, u'_1, \ldots, u'_{k_1})R).$$

(h) Delete the vertices v and w, and the edge between them from the graph G.

3. Repeat Steps 1) and 2) on each remaining edge with the DR label type.

Lemma 3.2.35. *Given an input of the chains graph G corresponding to a tree pair (D, R, t) representing an element α of V, Algorithm 3.2.34 terminates with a chains graph for which the only edges of label type DR are the ones for which the start vertex is the same as the end vertex. Moreover, the resulting graph still represents the same element α of V.*

Proof. We will first show that the process terminates. This is because each time we apply Step 2) we either eliminate a DR edge from further consideration, or we are diminishing number of edges of the graph G by at least 1. Recall that the number of edges is a finite number (initial size of E_G is m, where t is an m-tuple from the tree pair (D, R, t)). In the process we might or might not create new edges with labels DR, but this ultimately does not matter as the total number of edges to consider is always decreasing. Moreover, the algorithm continues if and only if there are edges with label type DR with distinct start and end vertex. Hence, when the algorithm stops, it produces a graph lacking such edges.

For the second part, we will look more closely what effect the edges replacements have on prefix substitution rules. Recall that we will be deleting the edge $e = (v, w, D(u_1, \ldots, u_k)R)$ for distinct vertices v and w. The prefix substitution rules given by this edge are:

$$(1)\begin{cases} (v)\alpha = u_1 & \\ (u_l)\alpha = u_{l+1} & \forall 1 \leq l < k \\ (u_k)\alpha = w & \end{cases}$$

Recall the process of augmentation from Algorithm 2.3.9. Recall the tree T defined in Algorithm 3.2.34. In Algorithm 3.2.34 we perform the augmentation of (D, R, t) by T at the leaves $v, u_1, \ldots, u_k$. We know that $L_T = \{p_i | 1 \leq i \leq l_1\} \cup \{q_j | 1 \leq j \leq l_2\} \cup \{r_h | 1 \leq h \leq l_3\}$. Then the prefix substitutions could be described by (1a),(1b) and (1c):

$$(1a)\begin{cases} \forall 1 \leq i \leq l_1 & \\ (vp_i)\alpha = u_1 p_i & \\ (u_l p_i)\alpha = u_{l+1} p_i & \forall 1 \leq l < k \\ (u_k p_i)\alpha = w p_i & \end{cases}$$

$$(1b)\begin{cases} \forall 1 \leq j \leq l_2 & \\ (vq_j)\alpha = u_1 q_j & \\ (u_l q_j)\alpha = u_{l+1} q_j & \forall 1 \leq l < k \\ (u_k q_j)\alpha = w q_j & \end{cases}$$

$$(1c)\begin{cases} \forall 1 \leq h \leq l_3 & \\ (vr_h)\alpha = u_1 r_h & \\ (u_l r_h)\alpha = u_{l+1} r_h & \forall 1 \leq l < k \\ (u_k r_h)\alpha = w r_h & \end{cases}$$

We could then cancel the edge e and represent these new prefix substitutions by a collection of following edges:

$$\{(vp_i, wp_i, D(u_1 p_i, \ldots, u_k p_i)R) | 1 \leq i \leq l_1\} \cup$$
$$\{(vq_j, wq_j, D(u_1 q_j, \ldots, u_k q_j)R) | 1 \leq j \leq l_2\} \cup$$
$$\{(vr_h, wr_h, D(u_1 r_h, \ldots, u_k r_h)R) | \forall 1 \leq h \leq l_3\}$$

We know that the labels on each of the new edges have to be of the type DR, because T is chosen so that it is a subtree of the tree R_v and hence the first letter of the label must be D, and T is also chosen to be a subtree of the tree D_w and hence the second letter of the label must be R. However, even after adjusting vertices, these edges would not form the chains graph corresponding to the new tree pair. This is because there would be vertices with an incoming edge with second letter of the label being R and with an outgoing edge with the first letter of the label being D. Algorithm 3.2.34 is inspired by the principle of concatenation of such neighbouring edges described in Remark 3.2.18. Let us analyse the effect of the suggested replacements of edges:

1. For each integer i such that $1 \leq i \leq l_1$, the edge

$$(u', v, (s')X(u'_1, \ldots, u'_{k_1})D(p_i))$$

is replaced by the edge

$$(u', wp_i, (s')X(u'_1, \ldots, u'_{k_1}, vp_i, u_1 p_i, \ldots, u_k p_i)R).$$

Let us analyse what it means in terms of prefix substitution rules. The initial edge encodes the following rules (2):

$$(2)\begin{cases}(u's')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \le l < k_1 \\ (u'_{k_1})\alpha = vp_i & \end{cases}$$

The resulting edge encodes the following rules (3):

$$(3)\begin{cases}\left.\begin{array}{rll}(u's')\alpha = u'_1 & & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \le l < k_1 & \\ (u'_{k_1})\alpha = vp_i & & \end{array}\right\}(2) \\ \left.\begin{array}{rll}(vp_i)\alpha = u_1p_i & & \\ (u_lp_i)\alpha = u_{l+1}p_i & \forall 1 \le l < k & \\ (u_kp_i)\alpha = wp_i & & \end{array}\right\}(1a)\end{cases}$$

Notice that for given p_i the rules (3) are a union of rules (2) and (1a). Hence, there has been no changes for α in this procedure. Note also that the vertex wp_i has been introduced as a new vertex of the graph and replaced appropriately in the edges which were starting from vertex w, with a proper descendant of wp_i being mapped as the beginning of the edge. Finally, the second letter of the label of the introduced edge is R, as p_i was not a leaf of the tree D_w, and hence the vertex wp_i becomes a range vertex of G.

2. For each integer j such that $1 \le j \le l_2$, the edge

$$(w, u'', (q_j)R(u''_1, \ldots, u''_{k_2})Y(s''))$$

is replaced by the edge

$$(vq_j, u'', D(u_1q_j, \ldots, u_kq_j, wq_j, u''_1, \ldots, u''_{k_2})Y(s'')).$$

Let us analyse what it means in terms of prefix substitution rules. The initial edge encodes the following rules (4):

$$(4)\begin{cases}(wq_j)\alpha = u''_1 & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \le l < k_2 \\ (u''_{k_2})\alpha = u''s'' & \end{cases}$$

The resulting edge encodes the following rules (5):

$$(5)\left\{\begin{array}{rll} (vq_j)\alpha = u_1q_j & & \\ (u_lq_j)\alpha = u_{l+1}q_j & \forall 1 \le l < k & \} (1b) \\ (u_kq_j)\alpha = wq_j & & \\ (wq_j)\alpha = u''_1 & & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \le l < k_2 & \} (4) \\ (u''_{k_2})\alpha = u''s'' & & \end{array}\right.$$

Notice that for given q_j the rules (5) are a union of rules (1b) and (4). Hence, there has been no change for α in this procedure. Note also that the vertex vq_j has been introduced as a new vertex of the graph and replaced appropriately in the edges which were ending at vertex v, with a proper descendant of vq_j being mapped to as the end of the edge. Finally, the first letter of the label of the edge introduced is D, as q_j was not a leaf of the tree R_v, and hence the vertex vq_j becomes a domain vertex of G.

3. For each integer h such that $1 \le h \le l_3$, the edges

$$e_1 = (u', v, (s')X(u'_1, \ldots, u'_{k_1})D(r_h))$$

and the edge

$$e_2 = (w, u'', (r_h)R(u''_1, \ldots, u''_{k_2})Y(s''))$$

are replaced by:

(a) If $e_1 \neq e_2$, by a single edge

$$\begin{aligned}(u', u'', (s')X(u'_1, \ldots, u'_{k_1}, vr_h, u_1r_h, \ldots,\\ \ldots, u_kr_h, wr_h, u''_1, \ldots, u''_{k_2})Y(s''))\end{aligned}$$

Let us analyse what it means in terms of prefix substitution rules. The two initial edges encodes the following rules (6) and (7):

$$(6)\left\{\begin{array}{ll}(u's')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \le l < k_1 \\ (u'_{k_1})\alpha = vr_h & \end{array}\right\}$$
$$(7)\left\{\begin{array}{ll}(wr_h)\alpha = u''_1 & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \le l < k_2 \\ (u''_{k_2})\alpha = u''s'' & \end{array}\right\}$$

The resulting edge encodes the following rules (8):

$$(8)\left\{\begin{array}{ll}(u's')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \le l < k_1 \\ (u'_{k_1})\alpha = vr_h & \end{array}\right\}(6)$$
$$\left\{\begin{array}{ll}(vr_h)\alpha = u_1 r_h & \\ (u_l r_h)\alpha = u_{l+1} r_h & \forall 1 \le l < k \\ (u_k r_h)\alpha = wr_h & \end{array}\right\}(1c)$$
$$\left\{\begin{array}{ll}(wr_h)\alpha = u''_1 & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \le l < k_2 \\ (u''_{k_2})\alpha = u''s'' & \end{array}\right\}(7)$$

Notice that for given r_h the rules (8) are a union of rules (6), (1c) and (7). Hence, there has been no change for α in this procedure.

(b) If $e_1 = e_2$, by a single edge

$$(vr_h, vr_h, D(u_1 r_h, \ldots, u_k r_h, wr_h, u'_1, \ldots, u'_{k_1})R)$$

Let us analyse what it means in terms of prefix substitution rules. The initial edge is then given by

$$(w, v, (r_h)R(u'_1, \ldots, u'_{k_1})D(r_h))$$

and it encodes the following rules (9):

$$(9)\left\{\begin{array}{ll}(wr_h)\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \le l < k_1 \\ (u'_{k_1})\alpha = vr_h & \end{array}\right.$$

The resulting edge encodes the following rules (10):

$$(10)\begin{cases} \left.\begin{array}{cc} (vr_h)\alpha = u_1 r_h & \\ (u_l r_h)\alpha = u_{l+1} r_h & \forall 1 \le l < k \\ (u_k r_h)\alpha = w r_h & \end{array}\right\}(1c) \\ \left.\begin{array}{cc} (wr_h)\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \le l < k_1 \\ (u'_{k_1})\alpha = v r_h & \end{array}\right\}(9) \end{cases}$$

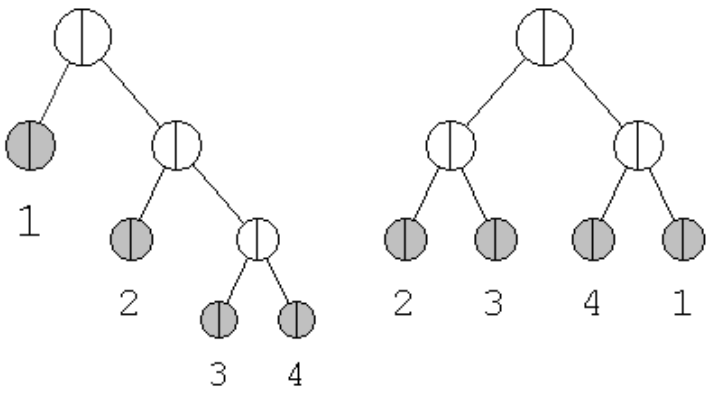
Notice that for given r_h the rules (10) are a union of rules (1c) and (9). Hence, there has been no changes for α in this procedure. Note also that the vertex vr_h has been introduced as a new vertex of the graph, and the label type of the new edge is DR, as it represents a new periodic orbit of leaves.

Note that this case illustrates how new periodic orbits of leaves are discovered in Algorithm 3.2.34.

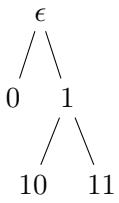
By these observations we can conclude that the resulting graph is indeed a chains graph of a new tree pair, which represents the same element α of V as the prefix substitution rules remain the same.

□

Example 3.2.36. Let us consider the following tree pair (D, R, t):



Let us construct the intersection $D \cap R$:



Hence, the set of vertices of the pre–chains graph is given by

$$\{0, 10, 11\}.$$

The set of edges of the pre–chains graph is given by

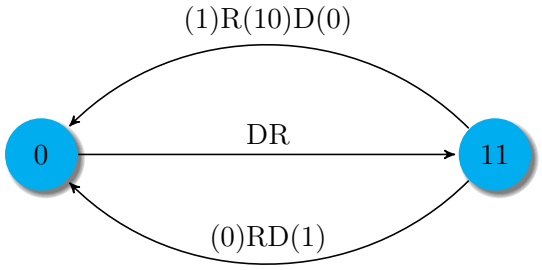
$$\{(0, 11, DR), (10, 0, DD(0)), (11, 0, (0)RD(1)), (11, 10, (1)RR)\}.$$

We will now transform the pre–chains graph into the chains graph G. There is only one neutral vertex of the pre–chains graph, and it is 10. It has precisely one incoming edge $(11, 10, (1)RR)$ and precisely one outgoing edge $(10, 0, DD(0))$, which we merge together to obtain the edge $(11, 0, (1)R(10)D(0))$.

Therefore, the vertex set of G is given by $V_G = \{0, 11\}$. Its set of edges is given by:

$$E_G = \{(0, 11, DR), (11, 0, (1)R(10)D(0)), (11, 0, (0)RD(1))\}$$

Hence the chains graph for (D, R, t) looks as follows:



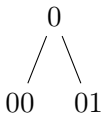
The Chains Graph of the Tree Pair (D, R, t)

We determine that there is precisely one edge of label type DR in E_G, and that its start vertex is distinct from its end vertex. Hence, we

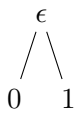
will apply Algorithm 3.2.34 to G.

1. We identify an appropriate edge with label type DR: $(0, 11, DR)$.

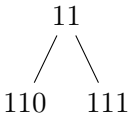
2. First of all, consider the component $(0)R_0$ of $R - D$. It is given by



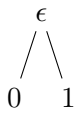
   Therefore, the tree R_0 is given by



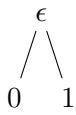
   Also, the component $(11)D_{11}$ of $D - R$ is given by



   Therefore, the tree D_{11} is given by



   The intersection $T = R_0 \cap D_{11}$ is also given by



   Therefore, we can identify the integers $l_1 = 0$ (number of leaves of T which are also leaves of R_0 but not leaves of D_{11}), $l_2 = 0$ (number of leaves of T which are also leaves of D_{11} but not leaves of R_0) and $l_3 = 2$ (number of leaves of T which are leaves of both D_{11} and R_0). Moreover, we identify the following addresses $r_1 = 0$ and $r_2 = 1$. Let us first consider $r_1 = 0$. The unique edge corresponding to r_1 entering the vertex 0 which we are looking for is $(11, 0, (1)R(10)D(0))$. The unique edge corresponding to r_1 leaving the vertex 11 which we are looking for is $(11, 0, (0)RD(1))$.

(f) As these are distinct edges, we replace them with a single edge

$$(11, 0, (1)R(10, 00, 110)D(1)).$$

Let us now consider $r_2 = 1$. The unique edge corresponding to r_2 entering the vertex 0 which we are looking for is

$$(11, 0, (1)R(10, 00, 110)D(1))$$

The unique edge corresponding to r_2 leaving the vertex 11 which we are looking for is also

$$(11, 0, (1)R(10, 00, 110)D(1)).$$

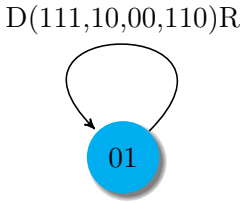
(g) As these are the same edge, we replace it with a single edge

$$(01, 01, D(111, 10, 00, 110)R)$$

and also introduce vertex 01 to the graph.

(g) We delete vertices 0 and 11 from the graph.

Hence we end up with a new graph G' such that $V_{G'} = \{01\}$ and $E_{G'} = \{(01, 01, D(111, 10, 00, 110)R)\}$, which looks as follows:



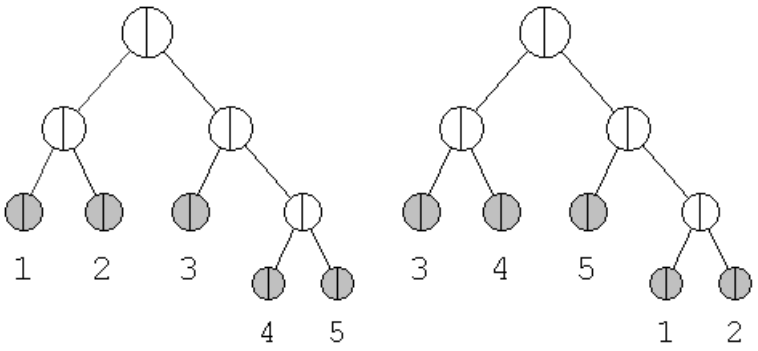
The Chains Graph G'

We want to find the tree pair corresponding to this chains graph. Leaf sets of both trees D and R are given by $L_D = L_R = \{00, 01, 10, 110, 111\}$.

We also have the following prefix substitutions:

$$(00)\alpha = 110$$
$$(01)\alpha = 111$$
$$(10)\alpha = 00$$
$$(110)\alpha = 01$$
$$(111)\alpha = 10$$

Therefore, the tree pair is:



Let us analyse what happened in Example 3.2.36. The chains graph of the resulting tree pair indicates that it can be interpreted as five leaves travelling on a single periodic orbit under the action of α. This fact is not immediately clear from the graph of the initial tree pair, and the reason for that is that the addresses 00 and 01 are simultaneously mapped to addresses 110 and 111 respectively, and this was initially expressed as a single map of address 0 to address 11, reflected by an edge with label type DR. However, the immediate preimage in terms of prefix substitution of the address 0 under the map α is not expressible as a single address. Similarly, the immediate image in terms of prefix substitution of the address 11 under the map α is not expressible as a single address. Hence, if we wanted to uncover the whole periodic orbit involved in this example, we had to perform simplification of the relevant DR edge, which was joining a range of sourcing and a domain of sinking. This is the general idea behind the Algorithm 3.2.34 which should give us intuition of what

happens in the algorithm and why.

Additionally, notice that in Example 3.2.36 with the application of the Algorithm 3.2.34 we reduced the number of carets in $D - R$ (and equivalently in $R - D$) by the number of carets present in the tree T.

Lemma 3.2.37. *Consider the chains graph G of a tree pair (D, R, t) representing an element α of V, such that G is the output of Algorithm 3.2.34. Let v be a domain vertex of the graph G. Then v has a unique outgoing edge, and the label type of this edge is DD.*

Proof. By Lemma 3.2.10 and Lemma 3.2.20, the vertex v has a unique outgoing edge e, and the first letter of the label type of this edge must be D, and so its label type is either DR or DD. However, if the label type of e was DR, then by Lemma 3.2.35 the end vertex of e would also need to be v. This is not possible, as by Lemma 3.2.10 and Lemma 3.2.20 the edge e cannot have an incoming edge with second letter of the label type being R. Hence, e must have label type DD.

□

Lemma 3.2.38. *Consider the chains graph G of a tree pair (D, R, t) representing an element α of V, such that G is the output of Algorithm 3.2.34. Let v be a range vertex of the graph G. Then v has a unique incoming edge, and the label type of this edge is RR.*

Proof. By Lemma 3.2.11 and Lemma 3.2.20, the vertex v has a unique incoming edge e, and the second letter of the label type of this edge must be R, and so its label type is either DR or RR. However, if the label type of e was DR, then by Lemma 3.2.35 the start vertex of e would also need to be v. This is not possible, as by Lemma 3.2.11 and Lemma 3.2.20 the edge e cannot have an outgoing edge with first letter of the label type being D. Hence, e must have label type RR.

□

The lemma below gives technical background which is later used in the Corollary 3.2.40 to prove that with Algorithm 3.2.34 we have uncovered all periodic orbits of leaves.

Lemma 3.2.39. *Consider the chains graph G of a tree pair (D, R, t) representing an element α of V, which is the output of Algorithm 3.2.34.*

Suppose that a word $s \in \{0, 1\}^*$ is such that some vertex of the chains graph G or some neutral leaf of the tree pair (D, R, t) present in the label

of a chain edge of G is a prefix v for s. This means that s underlies this vertex v of the graph or this neutral leaf v. Suppose also that there is a positive integer l such that for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^l = sw$. This means that some positive power of α fixes all words with some given finite prefix.

Then there must be an edge

$$(u_0, u_0, D(u_1, \ldots, u_k)R) \in E_G$$

for some vertex $u_0 \in V_G$, some non–negative integer k and some neutral leaves $u_1, \ldots, u_k$ of the tree pair (D, R, t), such that u_i if the prefix v of s for some integer $0 \leq i \leq k$.

Proof. Suppose that there are s, v and l as in the statement of this lemma. Suppose that there is an edge $(u_0, u_0, D(u_1, \ldots, u_k)R) \in E_G$ for some vertex $u_0 \in V_G$, some non–negative integer k and some neutral leaves $u_1, \ldots, u_k$ of (D, R, t), such that one of the leaves $u_0, u_1, \ldots, u_k$ is a prefix v of s. Then for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^{k+1} = sw$, and $k+1$ is the smallest positive integer satisfying this property, by the nature of edges in a chains graph. Hence $(k+1)|l$. As k is a non–negative integer, l is a positive integer as required. Hence, all the conditions of the lemma are met and this case can occur. We will show that this is the only possible scenario.

Suppose otherwise that s has a prefix which is a neutral leaf of a chains edge with the label type RR, RD or DD. By Lemma 3.2.35 the edges with label type DR represent the case above which we already considered.

First suppose that there is an edge

$$(u, u', (s^-)R(u'_1, \ldots, u'_{k'})Y(s^+)) \in E_G$$

for some vertices $u, u' \in V_G$, some positive integer k', some some neutral leaves $u'_1, \ldots, u'_{k'}$, some words $s^- \in \{0,1\}^+$ and $s^+ \in \{0,1\}^*$, and $Y \in \{D, R\}$. Suppose that $u'_{l'}$ is a prefix of s for some $l' \in \{1, \ldots, k'\}$. Hence there is a word $s' \in \{0,1\}^*$ such that $s = u'_{l'}s'$. Recall that for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^l = sw$. Then for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^{-l'} = (u'_{l'}s'w)\alpha^{-l'} = us^-s'w$. Also, $us^-s'w = (sw)\alpha^{-l'} = (sw)\alpha^l\alpha^{-l'} = (sw)\alpha^{-l'}\alpha^l = (us^-s'w)\alpha^l$. Hence, the word us^-s' is such that its prefix u is a vertex of the graph G and for all $w \in \{0,1\}^\omega$ we

have $us^-s'w = (us^-s'w)\alpha^l$.

Otherwise, suppose that there is an edge

$$(u, u', (s^-)X(u'_1, \ldots, u'_{k'})D(s^+)) \in E_G$$

for some vertices $u, u' \in V_G$, some positive integer k', some some neutral leaves $u'_1, \ldots, u'_{k'}$, some words $s^- \in \{0,1\}^+$ and $s^+ \in \{0,1\}^*$, and $X \in \{D, R\}$. Suppose that $u'_{l'}$ is a prefix of s for some $l' \in \{1, \ldots, k'\}$. Hence there is a word $s' \in \{0,1\}^*$ such that $s = u'_{l'}s'$. Recall that for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^l = sw$. Then for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^{k'+1-l'} = (u'_{l'}s'w)\alpha^{k'+1-l'} = u's^+s'w$. Also $u's^+s'w = (sw)\alpha^{k'+1-l'} = (sw)\alpha^l\alpha^{k'+1-l'} = (sw)\alpha^{k'+1-l'}\alpha^l = (u's^+s'w)\alpha^l$. Hence the word $u's^+s'$ is such that its prefix u' is a vertex of the graph G and for all $w \in \{0,1\}^\omega$ we have $u's^+s'w = (u's^+s'w)\alpha^l$.

Therefore, without loss of generality, we can presume that s has a prefix which is a vertex of the graph G.

Hence, suppose that s has a prefix which is a vertex u_0 of the graph G representing non-neutral leaf of the tree pair (D, R, t). By Corollary 3.2.22 vertex u_0 is either a leaf of the range tree but not the domain tree, or u_0 is a leaf of the domain tree but not the range tree.

Suppose that vertex u_0 is a leaf of the range tree but not the domain tree. Then by Lemma 3.2.38 the vertex u_0 has a unique incoming edge, and the label type of this edge is RR. Hence, the beginning of this edge is also a leaf of the range tree but not the domain tree. Hence for each such vertex u_0 we can build a unique sequence of vertices $(u_i)_{i\in\mathbb{N}_0}$, such that for all $i \in \mathbb{N}_0$ we have $(u_{i+1}, u_i, (\ldots)R(\ldots)R) \in E_G$. Now, note that $|V_G| < \infty$, so at some point we must have $u_j = u_{j+j'}$ for a positive j'. Choose minimal non–negative integer j such that $u_j = u_{j+j'}$ for some j', and then choose minimal positive j' such that $u_j = u_{j+j'}$. At first, we will cover the case when $j \neq 0$ and then when $j = 0$.

Consider the case when $j \neq 0$. That is, $u_0 \neq u_i$ for all positive integers i. Recall that vertices of the chains graph lie above pairwise disjoint subsets of $\mathfrak{C}$. Hence, for all $h \in \mathbb{Z}^-$ we have $\{u_0w | w \in \{0,1\}^\omega\} \cap \{(u_0w)\alpha^h | w \in \{0,1\}^\omega\} = \emptyset$. As α is a bijection on $\mathfrak{C}$ then for all $h \in \mathbb{Z}^-$ we have $\emptyset = (\emptyset)\alpha^{-h} = (\{u_0w | w \in \{0,1\}^\omega\} \cap \{(u_0w)\alpha^h | w \in \{0,1\}^\omega\})\alpha^{-h} = \{(u_0w)\alpha^{-h} | w \in \{0,1\}^\omega\} \cap \{(u_0w)\alpha^h\alpha^{-h} | w \in \{0,1\}^\omega\} = \{(u_0w)\alpha^{-h} | w \in \{0,1\}^\omega\} \cap \{u_0w | w \in \{0,1\}^\omega\}$. This collectively means that for all $g \in \mathbb{Z}\backslash\{0\}$ we have $\{u_0w | w \in \{0,1\}^\omega\} \cap \{(u_0w)\alpha^g | w \in$

$\{0,1\}^\omega\} = \emptyset$. Note that $\{sw|w \in \{0,1\}^\omega\} \subset \{u_0w|w \in \{0,1\}^\omega\}$. This contradicts the assumption that there is a positive integer l such that for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^l = sw$.

Consider the case when $j = 0$. This means that there is a positive integer j' such that $u_0 = u_{j'}$, and we consider the smallest such j'. This means that each of the vertices $u_1, \ldots, u_{j'-1}$ overlies a subset of $\mathfrak{C}$ which is disjoint from the subset underlying u_0. Now for each $i \in \{0, \ldots, j'-1\}$ we have an edge $(u_{i+1}, u_i, (s_{i+1})R(\ldots)R)$ for some words $s_{i+1} \in \{0,1\}^+$. For each $i \in \{0, \ldots, j'-1\}$ define a negative integer x_i which is given by $(-1-k)$, where k is the number of neutral leaves chained in the edge $(u_{i+1}, u_i, (s_{i+1})R(\ldots)R)$. Then for all $i \in \mathbb{N}_0$ and all words $w \in \{0,1\}^\omega$ we have $u_{i+1}s_{i+1}w = (u_iw)\alpha^{x_i}$. Notice that this implies that for all $i \in \mathbb{N}_0$ and for all $w \in \{0,1\}^\omega$ we have

$$u_{i+1}s_{i+1}s_i \ldots s_1w = (u_is_i \ldots s_1w)\alpha^{x_i} = (u_{i-1}s_{i-1} \ldots s_1w)\alpha^{x_{i-1}+x_i} = \\ \ldots = (u_0w)\alpha^{x_i+\ldots+x_1+x_0}$$

Hence, the only way we can possibly map a point u_0w to itself by a negative power of α is if we map it back to a point with prefix u_0. Define $X = \sum_{i=0}^{j'-1} x_i$ and $S = s_{j'}s_{j'-1} \ldots s_1$. If m is a negative integer and not a multiple of X, then intervals u_0w and $(u_0w)a^m$ are disjoint. Hence, we only need to consider powers of α^X. Notice that $(u_0w)\alpha^X = u_0Sw$. Hence, the only point of the form u_0w fixed by α^X is $u_0\overline{S}$. Similarly for all positive integers h,

$$(u_0w)\alpha^{hX} = u_0\underbrace{S \ldots S}_{h\text{-times}}w$$

the only point fixed by α^{hX} is $u_0\overline{S}$. Therefore, for each negative power of α, we either move all the points off the interval underlying u_0, or we fix at most one point. Now we come back the initial assumptions about s which has u_0 as its prefix. We know that there is a positive integer l such that for all $w \in \{0,1\}^\omega$ we have $sw = (sw)\alpha^l$. This implies that also $(sw)\alpha^{-l} = (sw)\alpha^l\alpha^{-l} = sw$. This creates a contradiction to the result. Hence, such an s cannot exist.

Otherwise suppose that vertex u_0 is a leaf of the domain tree but not the range tree. Then by Lemma 3.2.37 the vertex u_0 has a unique

outgoing edge, and the label type of this edge is DD. Hence, the end of this edge is also a leaf of the domain tree but not the range tree. Hence for each such vertex u_0 we can build a unique sequence of vertices $(u_i)_{i\in\mathbb{N}_0}$, such that for all $i \in \mathbb{N}_0$ we have $(u_i, u_{i+1}, D(\ldots)D(\ldots)) \in E_G$. Now, note that $|V_G| < \infty$, so at some point we must have $u_j = u_{j+j'}$ for a positive j'. Choose minimal non–negative integer j such that $u_j = u_{j+j'}$ for some j', and then choose minimal positive j' such that $u_j = u_{j+j'}$. At first, we will cover the case when $j \neq 0$ and then when $j = 0$.

Consider the case when $j \neq 0$. Recall that vertices of the chains graph lie above pairwise disjoint subsets of $\mathfrak{C}$. Hence, for all $h \in \mathbb{N}^+$ we have $\{u_0w|w \in \{0,1\}^\omega\} \cap \{(u_0w)\alpha^h|w \in \{0,1\}^\omega\} = \emptyset$. Note that $\{sw|w \in \{0,1\}^\omega\} \subset \{u_0w|w \in \{0,1\}^\omega\}$. This contradicts the assumption that there is a positive integer l such that for all $w \in \{0,1\}^\omega$ we have $(sw)\alpha^l = sw$.

Consider the case when $j = 0$. This means that there is a positive integer j' such that $u_0 = u_{j'}$, and we consider the smallest such j'. This means that each of the vertices $u_1, \ldots, u_{j'-1}$ overlies a subset of $\mathfrak{C}$ which is disjoint from the subset underlying u_0. Now for each $i \in \{0, \ldots, j'-1\}$ we have an edge $(u_i, u_{i+1}, D(\ldots)D(s_{i+1}))$ for some $s_{i+1} \in \{0,1\}^+$. For each $i \in \{0, \ldots, j'-1\}$ define a positive integer x_i which is given by $(1 + k)$ where k is the number of neutral leaves chained in the edge $(u_i, u_{i+1}, D(\ldots)D(s_{i+1}))$. Then for all $i \in \mathbb{N}_0$ and all words $w \in \{0,1\}^\omega$ we have $u_{i+1}s_{i+1}w = (u_iw)\alpha^{x_i}$. Notice that this implies that for all $i \in \mathbb{N}_0$ and for all $w \in \{0,1\}^\omega$ we have

$$u_{i+1}s_{i+1}s_i \ldots s_1w = (u_is_i \ldots s_1w)\alpha^{x_i} = (u_{i-1}s_{i-1} \ldots s_1w)\alpha^{x_{i-1}+x_i} = \ldots = (u_0w)\alpha^{x_0+\ldots+x_i}$$

Hence, the only way we can possibly map a point u_0w to itself by a positive power of α is if we map it back to a point with prefix u_0. Define $X = \sum_{i=0}^{j'-1} x_i$ and $S = s_{j'}s_{j'-1} \ldots s_1$. Hence, we only need to consider powers of α^X. Notice that $(u_0w)\alpha^X = u_0Sw$. Hence, the only point of the form u_0w fixed by α^X is $u_0\overline{S}$. Similarly for all positive integers h,

$$(u_0w)\alpha^{hX} = u_0\underbrace{S \ldots S}_{h\text{-times}}w$$

the only point fixed by α^{hX} is $u_0\overline{S}$. Therefore, for each positive power of α, we either move all the points off an interval underlying u_0, or we fix at most one point. Now we come back the initial assumptions about s which has u_0 as its prefix. We know that there is a positive integer l such that for all $w \in \{0,1\}^\omega$ we have $sw = (sw)\alpha^l$. This creates a contradiction to the result. Hence, such an s cannot exist.

Bringing all the conclusions together, our lemma gives the only possible result for given assumptions.

□

Corollary 3.2.40. *Consider a tree pair (D, R, t) and its chains graph G. After applying Algorithm 3.2.34 to G, we have identified all periodic orbits of leaves which will occur in a final outcome of the whole process presented in this section, namely resulting revealing pair. Each of these orbits occurs if and only if a single vertex with an edge starting and ending at itself with label type DR occurs in G.*

Proof. By Lemma 3.2.21 and Lemma 3.2.39. □

Note that with the lemma above, Algorithm 3.2.34 can be used on its own for the purpose of finding all periodic orbits of leaves, to test for their existence, or decide whether the element of Thompson's group V is torsion:

Corollary 3.2.41. *Consider a chains graph G which is the output of Algorithm 3.2.34 and which corresponds to a tree pair representing an element α of V. Then α is torsion if and only if the only label type of edges of G is DR.*

Proof. If the only label type of edges of G is DR, then α is trivially torsion. Conversely, if α is torsion then by Lemma 3.2.40 all of its periodic orbits of leaves have been identified and they are represented by edges of label type DR. □

The Algorithm Part III – Finding Attractors and Repellers

Consider a chains graph G which is the output of Algorithm 3.2.34 and which corresponds to the tree pair (D, R, t) representing element α of V. We will now present a series of four algorithms which will modify the given chains graph and result in another chains graph corresponding to another tree pair, which represents the same element α of V. The new

chains graph will continue to admit no edges with label type DR which do not represent full periodic orbits of its neutral leaves, as after Algorithm 3.2.34. Moreover, each edge with label type RR of the new chains graph will be guaranteed to admit a range of repulsion as its beginning and end vertex (which means that it locates a repeller), and each edge with label type DD of the new chains graph will be guaranteed to admit a domain of attraction as its beginning and end vertex (which means that it locates an attractor). It will be shown that these conditions are sufficient for the new chains graph to represent a revealing tree pair.

Finding Attractors

The first two algorithms aim to transform the graph G into a new chains graph where each edge with label type DD is guaranteed to admit a domain of attraction as its beginning and end vertex. Hence, it will indicate the location of an attractor.

It will be shown in Corollary 3.2.49 that any edge with label type DD and same start and end vertex represents an A–chain. However:

Lemma 3.2.42. *For a tree pair (D, R, t) representing $\alpha \in V$ and its chains graph G, an edge with label type DD with distinct start and end vertices corresponds to a DSS–chain.*

Proof. Suppose that $(v, w, D(u_1, \ldots, u_l)D(s))$ is an edge for some $v, w \in V_G$ such that $v \neq w$, some non-negative integer l, some word $s \in \{0,1\}^+$ and some neutral leaves $u_1, \ldots, u_l$ of the tree pair (D, R, t). We know by Lemma 3.2.22 that v and w cannot be neutral vertices of G. Hence both v and w are domain vertices. Recall that by Lemma 3.2.24 that this edge corresponds to a non-periodic leaf chain $(v, u_1, \ldots, u_l, ws)$. We are interested in what type of leaves v and ws are. As v is a domain vertex, it is a root of a component of $R - D$, and hence it can be either domain of attraction or domain of sinking leaf. But ws, the last leaf in the leaf chain starting at v, is not a proper descendant of v, hence v must be a domain of sinking. Now, as w is a domain vertex, ws is a leaf of a component of $R - D$, and hence it can be either an attractor or a sink. But v, the first leaf in the leaf chain ending at w, is not a proper ancestor of ws, hence w must be a sink.

As v is a domain of sinking and ws is a sink, by Definition 3.1.27 the

leaf chain $(v, u_1, \ldots, u_l, ws)$ corresponding to the edge

$$(v, w, D(u_1, \ldots, u_l)D(s))$$

is a DSS–chain. □

As we know that DSS–chains cannot occur in revealing tree pairs, Algorithms 3.2.44 and 3.2.51 are aimed at eliminating these edges.

The first algorithm presents an intermediate step, which eliminates *top vertices* (defined below) from the graph.

Definition 3.2.43 (Top vertex)**.** Consider a chains graph G which is an output of Algorithm 3.2.34 and which corresponds to the tree pair (D, R, t) representing element α of V. Let v be a domain vertex of G. If each of its incoming edges has label type RD, then we call v a *top vertex* of the graph G.

Algorithm 3.2.44. Consider a chains graph G which is an output of Algorithm 3.2.34 and which corresponds to the tree pair (D, R, t) representing element α of V.

1. Pick a top vertex v of the chains graph G. As it is a leaf of the domain tree D, by Lemma 3.2.37 it has exactly one outgoing edge, and this edge has label type DD. Say that this edge ends at a vertex w. Then $(v, w, D(\ldots)D(\ldots)) \in E_G$. Note that as w is an end vertex of an edge of the label type DD, w is also a domain vertex.

   Let the edge between v and w be given by

   $$(v, w, D(u_1, \ldots, u_k)D(s))$$

   for some non–negative integer k, some neutral leaves $u_1, \ldots, u_k$ of the tree pair (D, R, t) and some word $s \in \{0, 1\}^+$.

   Consider each edge finishing at v. By definition of top vertex, it needs to be of the form $(u, v, (s'')R(u'_1, u'_2, \ldots, u'_{k'})D(s'))$ for some vertex $u \in V_G$, some non-negative integer k', some neutral leaves $u'_1, \ldots, u'_{k'}$ of the tree pair (D, R, t) and some words $s', s'' \in \{0, 1\}^+$. As v is a domain vertex, at least two such edges exist by Lemma 3.2.10 and Lemma 3.2.20.

   Let us now perform the following process:

(a) Replace each such edge

$$(u, v, (s'')R(u'_1, u'_2, \ldots, u'_{k'})D(s'))$$

with the edge

$$(u, w, (s'')R(u'_1, u'_2, \ldots, u'_{k'}, vs', u_1s', \ldots, u_ks')D(ss')).$$

(b) Delete the vertex v and the edge $(v, w, D(u_1, \ldots, u_k)D(s))$.

2. Repeat Step 1) until there are no more top vertices left.

Lemma 3.2.45. *Consider a chains graph which is an output of Algorithm 3.2.34 which we input into Algorithm 3.2.44. Then, the algorithm terminates with the chains graph G which admits no top vertices. Moreover, the resulting graph still represents the same element α of V.*

Proof. Note that with each cancellation of a top vertex, the vertex w might or might nor turn into a top vertex. However, the total number of vertices of the chains graph goes down by one in each iteration of the process, and hence the algorithm has to terminate. As termination is marked by the point when the chains graph admits no more top vertices, we have proven the first part of the claim.

For the second part, we will look more closely at what effect the edges replacements have on prefix substitution rules. Recall that we will be deleting the edge $e = (v, w, D(u_1, \ldots, u_k)D(s))$ for distinct vertices v and w, where v is a top vertex. The prefix substitution rules given by this edge are:

$$(1)\begin{cases} (v)\alpha = u_1 & \\ (u_l)\alpha = u_{l+1} & \forall 1 \leq l < k \\ (u_k)\alpha = ws & \end{cases}$$

Let us consider what it would mean for the prefix substitution rules if without changing α instead of having one whole leaf v travel to the leaf ws under the action of powers of α, we split the leaf v further into smaller pieces, namely introduce a tree rooted at the leaf v of the tree D of the tree pair (D, R, t), at all neutral leaves $u_1, \ldots, u_k$ of both trees D and R, and at the leaf ws of the tree R.

An example of a tree which we could introduce could be the tree R_v rooted at the node v of the tree R. Then the prefix substitutions could

be equally described by (1'):

$$(1')\begin{cases} \forall s' \in L_{R_v} & \\ (vs')\alpha = u_1 s' & \\ (u_l s')\alpha = u_{l+1} s' & \forall 1 \leq l < k \\ (u_k s')\alpha = wss' & \end{cases}$$

We could then cancel the edge e and represent these new prefix substitutions by a collection of following edges:

$$\{(vs', w, D(u_1 s', \ldots, u_k s')D(ss'))|s' \in L_{R_v}\}$$

We know that the label on each of the new edges has to be of type DD, because R_v is chosen depending on the targets of incoming edges of v. Also, the operation does not impact the fact that w is still a domain vertex of the graph. However, even after adjusting vertices, these edges would not form the chains graph corresponding to the new tree pair. This is because there would be vertices with an incoming edge with second letter of the label being R (v is replaced by vertices vs', and edges coming into v change the second letter $D(s')$ of their labels to R) and with an outgoing edge with the first letter of the label being D. Moreover, vs' is a neutral vertex of the new tree pair, for all $s' \in L_{R_v}$. Algorithm 3.2.44 is inspired by the principle of concatenation of such neighbouring edges described in Remark 3.2.18. Let us analyse the effect of the suggested replacements of edges:

For each word $s' \in L_{R_v}$, the edge

$$(u, v, (s'')R(u'_1, u'_2, \ldots, u'_{k'})D(s'))$$

is replaced by the edge

$$(u, w, (s'')R(u'_1, u'_2, \ldots, u'_{k'}, vs', u_1 s', \ldots, u_k s')D(ss')).$$

Let us analyse what it means in terms of prefix substitution rules. The initial edge encodes the following rules (2):

$$(2)\begin{cases} (us'')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \leq l < k' \\ (u'_{k'})\alpha = vs' & \end{cases}$$

The resulting edge encodes the following rules (3):

$$(3)\begin{cases} \left.\begin{array}{cc} (us'')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \leq l < k' \\ (u'_{k'})\alpha = vs' & \end{array}\right\}(2) \\ \left.\begin{array}{cc} (vs')\alpha = u_1 s' & \\ (u_l s')\alpha = u_{l+1} s' & \forall 1 \leq l < k \\ (u_k s')\alpha = wss' & \end{array}\right\}(1') \end{cases}$$

Notice that for given s' the rules (3) are a union of rules (2) and $(1')$. Hence, there has been no change to α in this procedure.

From this observation we can conclude that the resulting graph is indeed the chains graph of a new tree pair, which represents the same element α of V, as the prefix substitution rules remain the same.

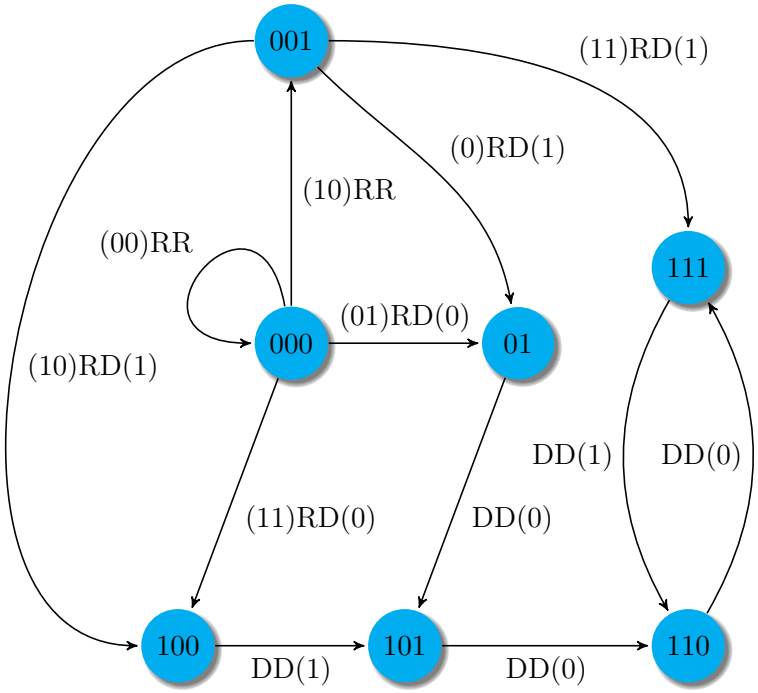
□

It is also important to realise that application of Algorithm 3.2.44 does not cancel the benefits of Algorithm 3.2.34:

Lemma 3.2.46. *Consider a chains graph G which is output by Algorithm 3.2.34 and Algorithm 3.2.44 applied one after another. Then G remains unchanged by a subsequent application of Algorithm 3.2.34, and so all the properties of a graph which is output by Algorithm 3.2.34 are preserved.*

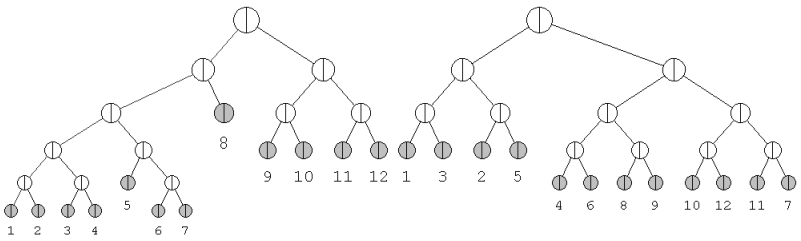
Proof. The start vertex of each edge with label type DR of graph output by Algorithm 3.2.34 is also the end vertex of that edge. Algorithm 3.2.44 neither cancels nor introduces any edges with label type DR. Hence, if the graph G is an output of Algorithm 3.2.34 and Algorithm 3.2.44 applied one after another, it is still the case that beginning vertex of each edge with label type DR is also the end vertex of that edge. Therefore, the graph G remains unchanged under potential application of Algorithm 3.2.34. □

Example 3.2.47. Consider the chains graph G given in the picture below:



The Chains Graph G

The graph G corresponds to the following tree pair (D, R, t):

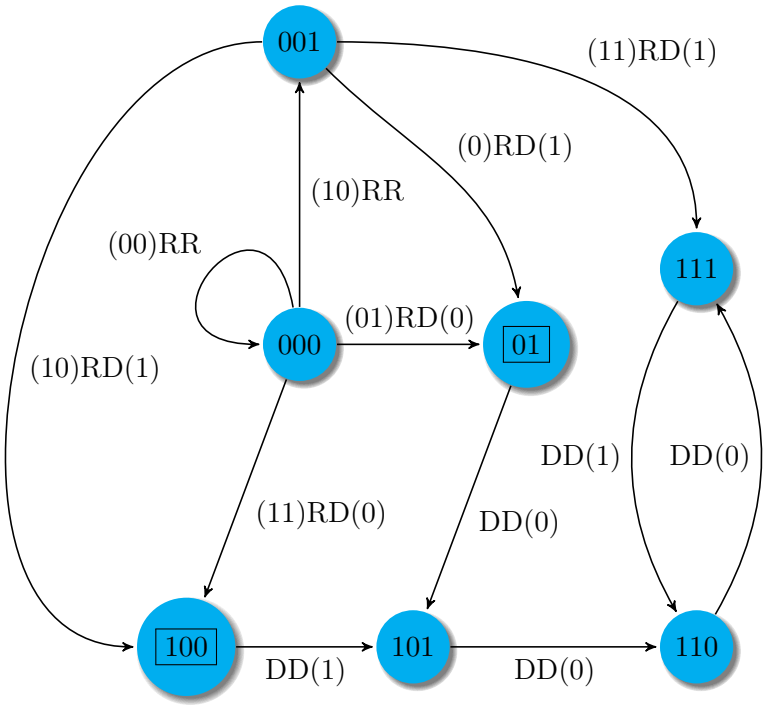


We can read off the set of vertices of G and the set of edges of G from the chains graph picture:

$$V_G = \{000, 001, 01, 100, 101, 110, 111\}$$

$$E_G = \{(000, 000, (00)RR), (000, 01, (01)RD(0)), (000, 001, (10)RR,),$$
$$(000, 100, (11)RD(0)), (001, 01, (0)RD(1)), (001, 100, (10)RD(1)),$$
$$(001, 111, (11)RD(1)), (01, 101, DD(0)), (100, 101, DD(1)),$$
$$(101, 110, DD(0)), (110, 111, DD(0)), (111, 110, DD(1))\}$$

We notice that the graph G has no edges with label type DR (and in particular no edges with label type DR with distinct beginning and end vertices). Hence by Lemma 3.2.35 it remains unchanged by Algorithm 3.2.34. We can therefore input it into Algorithm 3.2.44. Therefore, we analyse the graph in order to identify top vertices. To start with, the only candidates for top vertices are domain vertices, namely vertices from the set $\{01, 100, 101, 110, 111\}$. Now, we can see that each vertex from the set
$\{101, 110, 111\}$ has an incoming edge with label type DD. Hence we analyse the remaining vertices from the set $\{01, 100\}$ and conclude that all of their incoming edges are of the label type RD. Thus, we conclude that they are both top vertices.



The Chains Graph G with Top Vertices Indicated

We now proceed with the application of Algorithm 3.2.44 to each of these vertices:

1. Consider the vertex 01.

   (a) It has one outgoing edge $(01, 101, DD(0))$.

   (b) It has two incoming edges, namely $(000, 01, (01)RD(0))$ and $(001, 01, (0)RD(1))$.

   (c) We replace the edge $(000, 01, (01)RD(0))$ with the edge $(000, 101, (01)R(010)D(00))$.

   (d) We replace the edge $(001, 01, (0)RD(1))$ with the edge $(001, 101, (0)R(011)D(01))$.

   (e) We delete the vertex 01 and the edge $(01, 101, DD(0))$.

2. Consider the vertex 100.

   (a) It has one outgoing edge $(100, 101, DD(1))$.

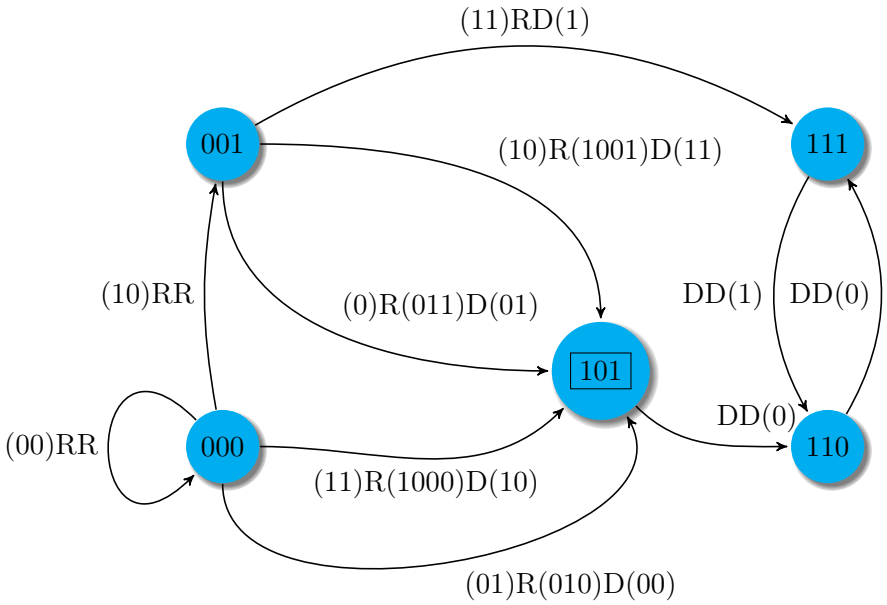
   (b) It has two incoming edges, namely $(000, 100, (11)RD(0))$ and $(001, 100, (10)RD(1))$.

   (c) We replace the edge $(000, 100, (11)RD(0))$ with the edge $(000, 101, (11)R(1000)D(10))$.

   (d) We replace the edge $(001, 100, (10)RD(1))$ with the edge $(001, 101, (10)R(1001)D(11))$.

   (e) We delete the vertex 100 and the edge $(110, 101, DD(1))$.

At this point, the intermediate chains graph looks as follows, with its only top vertex indicated:



The New Chains Graph with the Top Vertex Indicated

The vertex 101 is now a top vertex, and the algorithm needs to be applied to it:

1. The vertex 101 has precisely one outgoing edge $(101, 110, DD(0))$.

2. There are four incoming edges, namely

$$(000, 101, (01)R(010)D(00))$$
$$(000, 101, (11)R(1000)D(10))$$
$$(001, 101, (0)R(011)D(01))$$
$$(001, 101, (10)R(1001)D(11))$$

3. We replace the edge $(000, 101, (01)R(010)D(00))$ with the edge

$$(000, 110, (01)R(010, 10100)D(000)).$$

4. We replace the edge $(000, 101, (11)R(1000)D(10))$ with the edge

$$(000, 110, (11)R(1000, 10110)D(010)).$$

5. We replace the edge $(001, 101, (0)R(011)D(01))$ with the edge

$$(001, 110, (0)R(011, 10101)D(001)).$$

6. We replace the edge $(001, 101, (10)R(1001)D(11))$ with the edge

$$(001, 110, (10)R(1001, 10111)D(011)).$$

7. We delete the vertex 101 and the edge $(101, 110, DD(0))$.

At this point the chains graph admits only two domain vertices 110 and 111, but each of them has an ingoing edge with the label type DD, and hence none of them is a top vertex. Hence, Algorithm 3.2.44 terminates. It terminates with the chains graph G with the vertices set given by $V_G = \{000, 001, 110, 111\}$ and edges set given by:

$$\begin{aligned} E_G = \{&(000, 000, (00)RR), (000, 110, (01)R(010, 10100)D(000)), \\ &(000, 001, (10)RR,), (000, 110, (11)R(1000, 10110)D(010)), \\ &(001, 110, (0)R(011, 10101)D(001)), \\ &(001, 110, (10)R(1001, 10111)D(011)), (001, 111, (11)RD(1)), \\ &(110, 111, DD(0)), (111, 110, DD(1))\} \end{aligned}$$

We could interpret Algorithm 3.2.44 as prolonging appropriate edges with label type RD towards potential attractor(s). However, in the example above we do not detect any attractors by the following lemma:

Lemma 3.2.48. *Consider a chains graph G corresponding to the tree pair (D, R, t) representing an element α of Thompson's group V. Consider a domain vertex v of G and its unique outgoing edge*

$$(v, w, D(u_1, \ldots, u_k)D(s))$$

for some vertex w of G, some non-negative integer k, some neutral leaves $u_1, \ldots, u_k$ of the tree pair (D, R, t) and some word $s \in \{0, 1\}^*$. Then $v = w$ if and only if v is a domain of attraction leaf of the tree D and vs is an attractor leaf of the tree R.

Proof. Suppose that $v = w$. Consider the edge $(v, v, D(u_1, \ldots, u_k)D(s))$

of the graph G. It indicates that α admits the following prefix replacements for all positive integers i such that $1 \leq i < k$:

$$\begin{aligned}(v)\alpha &= u_1\\ (u_i)\alpha &= u_{i+1}\\ (u_k)\alpha &= vs\end{aligned}$$

By Definition 3.1.10 and Definition 3.1.11, v is a domain of attraction leaf of the tree D and vs is an attractor leaf of the tree R.

Now, suppose that v is a domain of attraction leaf of D. Then there is a unique sequence of leaves $v, (v)\alpha, (v)\alpha^2, \ldots, (v)\alpha^l$ of the tree pair (D, R, t) for some positive integer l, such that $(v)\alpha, (v)\alpha^2, \ldots, (v)\alpha^{l-1}$ are neutral leaves of the tree pair (D, R, t), $(v)\alpha^l$ is a leaf of range tree R but not domain tree D, and $(v)\alpha^l$ is a proper descendant of v. This is reflected in the graph G by existence of an edge $(v, v, D((v)\alpha, (v)\alpha^2, \ldots, (v)\alpha^{l-1})D(s'))$ for some non-empty word s' such that $vs' = (v)\alpha^l$. But v has a unique outgoing edge, and so $w = v$, $k = l - 1$, for all positive integers i such that $1 \leq i < k$ we have $u_i = (v)\alpha^i$ and $s = s'$. Note that this implies that vs is an attractor.

□

Corollary 3.2.49. *Consider a chains graph G corresponding to the tree pair (D, R, t) representing an element α of Thompson's group V. Suppose that there is an edge e of the label type DD starting and finishing at the same vertex. Then, the edge e corresponds to an A–chain of leaves.*

Proof. By Lemma 3.2.48, Lemma 3.2.24 and Definition 3.1.26. □

The second algorithm transforms a chains graph which is an output of Algorithm 3.2.34 and Algorithm 3.2.44 applied one after another into a new chains graph where each edge with label type DD is guaranteed to admit a domain of attraction as its beginning and end vertex, and hence locates an attractor. Before we present this algorithm, we will prove the following lemma:

Lemma 3.2.50. *Consider a chains graph G which is an output of Algorithm 3.2.34 and Algorithm 3.2.44 applied one after another. Then, each domain vertex has a unique incoming edge with label type DD. Moreover, each domain vertex v_1 is a member of a unique set of domain vertices*

$\{v_1, \ldots, v_j\}$ for some positive integer j, such that for all integers i such that $1 \leq i < j$ there is an edge $(v_i, v_{i+1}, D(...)D(...)) \in E_G$ and also $(v_j, v_1, D(...)D(...)) \in E_G$.

Proof. We will first show that there is a bijection between the set of domain vertices and the set of edges with label type DD. Then we will show that given that no top vertices are allowed, the conclusions of the lemma must hold.

As by Lemma 3.2.46 the graph G remains unchanged by Algorithm 3.2.34, by Lemma 3.2.37 we know that each domain vertex admits precisely one outgoing edge, and the label type of this edge is DD. The beginning of an edge with label type DD cannot be a range vertex, and by Lemma 3.2.21 it also cannot be a neutral vertex (as neutral vertices of chains graph are only at the beginnings and ends of edges of label type DR). Hence, the beginning vertex of an edge with label type DD is always a domain vertex. Note also that beginning vertex of any edge is unique. Hence, there is exactly same number of domain vertices as edges with label type DD in the graph G.

Now notice that by Lemma 3.2.45 none of the domain vertices can be a top vertex and therefore by Lemma 3.2.10 and Lemma 3.2.20 each of domain vertices must have at least one incoming edge with label type DD. But each edge with label type DD has unique end vertex which is a domain vertex. Hence as there is exactly same number of domain vertices as edges with label type DD, by the pigeonhole principle, each domain vertex must admit precisely one incoming edge with label type DD.

Finally, let us consider a domain vertex v_1. It has precisely one outgoing edge, and the label type of this edge is DD. Hence we can build unique sequence $(v_i)_{\mathbb{N}^+}$ such that the graph G admits an edge $(v_i, v_{i+1}, D(\ldots)D(\ldots))$. However, as $|V_G| < \infty$, there will be repetitions of vertices in the sequence. For any given v_i which occurs in the sequence for the first time, namely $v_i \neq v_k$ for any positive integer k such that $k < i$, the vertex v_{i+1} can either be a previously unlisted vertex, or it can be v_1, as all the other listed vertices $v_2, v_3, \ldots, v_i$ already have their unique incoming edge with label type DD identified. As $|V_G| < \infty$, we must at some point finish at the vertex v_1, say $v_{j+1} = v_1$ for some positive integer j. When we close the loop, all of the vertices involved in the loop already have their unique incoming and outgoing edges of type DD identified, so they cannot be used to build any other sequences with

starting vertices outside the set $\{v_1, v_2, \ldots, v_j\}$.

Therefore, we have proven the lemma. □

We can partition the domain vertices into sets belonging to loops joined by edges with label type DD.

Now we are prepared to proceed with the algorithm after which we will have all the attractors which we need for a given α of V:

Algorithm 3.2.51. Consider a chains graph G which is an output of Algorithm 3.2.34 and Algorithm 3.2.44 applied one after another. Suppose that G corresponds to the tree pair (D, R, t) representing element α of V.

1. Pick a domain vertex v such that its unique outgoing edge with label type DD does not end at v. This vertex will remain in the graph.

   Identify the unique edge of label type DD leaving the vertex v, say $(v, u, D(u_1, u_2, \ldots, u_k)D(s))$ for some vertex $u \in V_G$, some non-negative integer k, some neutral leaves $u_1, u_2, \ldots, u_k$ of the tree pair (D, R, t) and some word $s \in \{0,1\}^+$.

   Identify the unique edge of label type DD leaving the vertex u, say $(u, w, D(u'_1, u'_2, \ldots, u'_{k'})D(s'))$ for some vertex $w \in V_G$, some non–negative integer k', some neutral leaves $u'_1, u'_2, \ldots, u'_{k'}$ of the tree pair (D, R, t) and some word $s' \in \{0,1\}^+$.

   Then perform the following action:

   (a) Replace the edge $(v, u, D(u_1, u_2, \ldots, u_k)D(s))$ with the edge

   $$(v, w, D(u_1, u_2, \ldots, u_k, us, u'_1 s, u'_2 s, \ldots, u'_{k'} s)D(s's)).$$

   Consider each other edge finishing at u. By Lemma 3.2.50 it must be of the form $(u', u, (s''')R(u''_1, u''_2, \ldots, u''_{k''})D(s''))$ for some vertex u', some non-negative integer k'', some neutral leaves $u''_1, \ldots, u''_{k''}$ of the tree pair (D, R, t) and some words $s'', s''' \in \{0,1\}^+$.

   (b) Replace each such edge with the edge

   $$\begin{aligned}(u', w, (s''')R(u''_1, u''_2, \ldots,\\ \ldots, u''_{k''}, us'', u'_1 s'', u'_2 s'', \ldots,\\ \ldots, u'_{k'} s'')D(s's''))\end{aligned}$$

(c) Delete the vertex u and the edge

$$(u, w, D(u'_1, u'_2, \ldots, u'_{k'})D(s')).$$

(d) Repeat the process until the unique edge with the label type DD leaving the vertex v also ends at v.

2. Repeat Step 1) until all domain vertices are at the same time beginning and end vertices of their unique outgoing edges.

Lemma 3.2.52. *Consider a chains graph G which is an output of Algorithm 3.2.34, Algorithm 3.2.44 and Algorithm 3.2.51 applied one after another. Suppose that G corresponds to the tree pair (D, R, t) representing element α of V.*

Then the algorithm terminates with a new chains graph, with each of its domain vertices corresponding to a domain of attraction leaf, and hence locating an attractor.

Moreover, each connected component of $R - D$ contains an attractor.

Also, the resulting graph still represents the same element α of V.

Proof. The algorithm terminates as each of its iterations reduces total number of vertices by 1. The algorithm is defined to terminate when each domain vertex is at the same time beginning and end of its unique outgoing edge. By Lemma 3.2.48 this means that each of the domain vertices v of G corresponds to a domain of attraction leaf, and hence locates an attractor vs for some non-empty word s.

Moreover, connected components of $R - D$ are in bijection with domain vertices of G, as each of the connected components of $R - D$ is rooted at a domain vertex. By Lemma 3.2.48 vs is an attractor and it's located at a leaf of the connected component rooted at v. As this holds for each domain vertex v, we conclude that each connected component of $R - D$ contains an attractor.

For the second part, we will look more closely at what effect the edge replacements have on prefix substitution rules. Recall that we will be deleting the edge $e = (u, w, D(u'_1, u'_2, \ldots, u'_{k'})D(s'))$ for distinct vertices v and w. The prefix substitution rules given by this edge are:

$$(1) \begin{cases} (u)\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \leq l < k' \\ (u'_{k'})\alpha = ws' & \end{cases}$$

Let us consider what it would mean for the prefix substitution rules if, without changing α, instead of having one whole leaf u travel to the leaf ws' under the action of powers of α, we split the leaf u further into smaller pieces, introducing a tree rooted at the leaf u of the tree D of the tree pair (D, R, t), at all neutral leaves $u_1, \ldots, u_k$ of both trees D and R, and at the leaf ws' of the tree R.

An example of a tree which we could introduce could be the tree R_u rooted at the node u of the tree R. Then the prefix substitutions could be equally described by $(1')$:

$$(1') \begin{cases} \forall s'' \in L_{R_u} & \\ (us'')\alpha = u'_1 s'' & \\ (u'_l s'')\alpha = u'_{l+1} s'' & \forall 1 \leq l < k' \\ (u'_{k'} s'')\alpha = ws's'' & \end{cases}$$

We could then cancel the edge e and represent these new prefix substitutions by the collection of following edges:

$$\{(us'', w, D(u_1 s'', \ldots, u_k s'')D(s's''))|s'' \in L_{R_u}\}$$

We know that the labels on each of the new edges have to be of the type DD, because R_u is chosen depending on the targets of incoming edges of u. Also, w is still a domain vertex of the new chains graph. However, even after adjusting vertices, these edges would not form the chains graph corresponding to the new tree pair. This is because there would be vertices with an incoming edge with second letter of the label being R and with an outgoing edge with the first letter of the label being D. Algorithm 3.2.51 is inspired by the principle of concatenation of such neighbouring edges described in Remark 3.2.18. Let us analyse the effect of the suggested replacements of edges:

1. For each word $s'' \in L_{R_u} \backslash \{s\}$, the edge

$$(u', u, (s''')R(u''_1, u''_2, \ldots, u''_{k''})D(s''))$$

is replaced by the edge

$$(u', w, (s''')R(u''_1, u''_2, \ldots, u''_{k''}, us'', u_1 s'', \ldots, u_k s'')D(s's'')).$$

Let us analyse what it means in terms of prefix substitution rules.

The initial edge encodes the following rules (2):

$$(2)\begin{cases}(u's''')\alpha = u''_1 & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \le l < k'' \\ (u''_{k''})\alpha = us'' & \end{cases}$$

The resulting edge encodes the following rules (3):

$$(3)\begin{cases}\left.\begin{array}{ll}(u's''')\alpha = u''_1 & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \le l < k'' \\ (u''_{k''})\alpha = us'' & \end{array}\right\}(2) \\ \left.\begin{array}{ll}(us'')\alpha = u'_1 s'' & \\ (u'_l s'')\alpha = u'_{l+1} s'' & \forall 1 \le l < k' \\ (u'_{k'} s'')\alpha = ws's'' & \end{array}\right\}(1')\end{cases}$$

Notice that for given s'' the rules (3) are a union of rules (2) and $(1')$. Hence, there has been no change for α in this procedure.

2. The edge

$$(v, u, D(u_1, u_2, \ldots, u_k)D(s))$$

is replaced with the edge

$$(v, w, D(u_1, u_2, \ldots, u_k, us, u'_1 s, u'_2 s, \ldots, u'_{k'} s)D(s's))$$

Let us analyse what it means in terms of prefix substitution rules. The initial edge encodes the following rules (4):

$$(4)\begin{cases}(v)\alpha = u_1 & \\ (u_l)\alpha = u_{l+1} & \forall 1 \le l < k \\ (u_k)\alpha = us & \end{cases}$$

The resulting edge encodes the following rules (5):

$$(5)\begin{cases}\left.\begin{array}{ll}(v)\alpha = u_1 & \\ (u_l)\alpha = u_{l+1} & \forall 1 \le l < k \\ (u_k)\alpha = us & \end{array}\right\}(4) \\ \left.\begin{array}{ll}(us)\alpha = u'_1 s & \\ (u'_l s)\alpha = u'_{l+1} s & \forall 1 \le l < k' \\ (u'_{k'} s)\alpha = ws's & \end{array}\right\}(1')\end{cases}$$

Notice that for $s'' = s$ the rules (5) are a union of rules (4) and (1′). Hence, there has been no change for α in this procedure.

By this observation we can conclude that the resulting graph is indeed a chains graph of a new tree pair, which represents the same element α of V, as the prefix substitution rules remain the same.

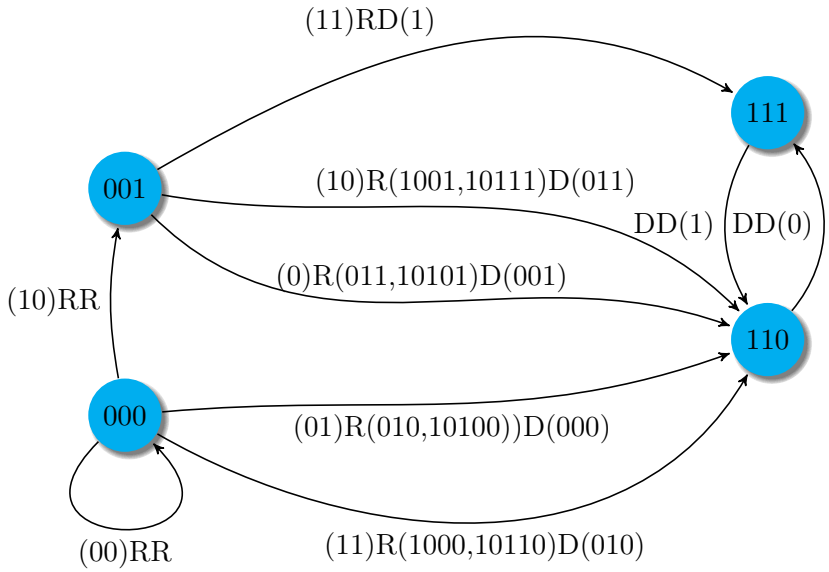
□

It is also important to realise that application of Algorithm 3.2.51 does not cancel the benefits of Algorithm 3.2.34 and Algorithm 3.2.44:

Lemma 3.2.53. *Consider a chains graph G which is an output of Algorithm 3.2.34, Algorithm 3.2.44 and Algorithm 3.2.51 applied one after another. Then G remains unchanged by a subsequent applications of Algorithm 3.2.34 and Algorithm 3.2.44, and so it keeps all the properties of a graph which is an output of Algorithm 3.2.34 and of a graph which is an output of Algorithm 3.2.44.*

Proof. The start vertex of each edge with label type DR of an output graph of Algorithm 3.2.34 is also the end vertex of that edge. Each of Algorithm 3.2.44 and Algorithm 3.2.51 neither cancels nor introduces any edges with label type DR. Hence, if the graph G is an output of Algorithm 3.2.34, Algorithm 3.2.44 and Algorithm 3.2.51 applied one after another, it is still the case that beginning vertex of each edge with label type DR is also the end vertex of that edge. Therefore, the graph G remains unchanged under potential application of Algorithm 3.2.34. Moreover, by Lemma 3.2.48 Lemma 3.2.52, each domain vertex of the graph G is an end vertex of an edge of the label type DD, and hence it is not a top vertex. Hence, if the graph G is an output of Algorithm 3.2.34, Algorithm 3.2.44 and Algorithm 3.2.51 applied one after another, it is still the case that it admits no top vertices. Therefore, the graph G remains unchanged under potential application of Algorithm 3.2.44. □

Example 3.2.54. Consider the chains graph G given in the picture below:



The Chains Graph G

Notice that G is a final chains graph from Example 3.2.47, and hence an output of Algorithm 3.2.34 and Algorithm 3.2.44 applied one after another. We can therefore input it into Algorithm 3.2.51 in order to create all possible attractors. To start with, we notice that neither of the domain vertices 110 and 111 have an edge starting and ending at it. Hence, we can pick any of them for application of the algorithm. Let us pick the vertex 110.

1. The edge $(110, 111, DD(0))$ is the unique outgoing edge of the vertex 110.

2. The edge $(111, 110, DD(1))$ is the unique outgoing edge of the vertex 111.

3. We replace the edge $(110, 111, DD(0))$ with the edge

$$(110, 110, D(1110)D(10)).$$

4. There is one more incoming edge of the vertex 111, namely

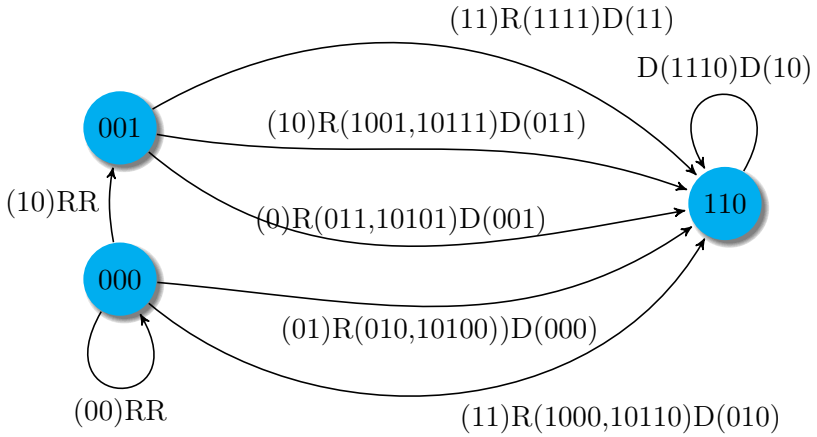
$$(001, 111, (11)RD(1)).$$

We replace it with the edge

$$(001, 110, (11)R(1111)D(11)).$$

5. We delete the vertex 111 and the edge $(111, 110, DD(1))$.

At this point the new chains graph looks as follows:



The New Chains Graph

We notice that there is only one domain vertex 110 left, and it has the edge $(110, 110, D(1110)D(10))$ beginning and ending at it. Hence, the algorithm terminates here. We can deduce that 110 is the unique domain of attraction leaf of the new tree D, and 11010 is the unique attractor leaf of the new range tree R.

Notice that we could have picked the vertex 111 to stay in the graph, as opposed to vertex 110, and we would still obtain the desired results. In this case an informal motivation for picking 110 to stay was that there were only two edges finishing at vertex 111 as opposed to five edges finishing at the vertex 110, and the lengths of the edges from 110 to 111 and from 111 to 110 had the same number of neutral vertices involved, namely zero. This means that picking the vertex 110 produced one more caret on each of the trees D and R, while picking vertex 111 would produce four more carets on each of the trees D and R.

In general, depending on our needs, we might have different definitions of best choice of the vertex which we pick. It might be for instance the

minimal number of new neutral leaves induced, the minimal depth of the resulting tree pair, or some other condition.

Finding Repellers

The last two algorithms aim to transform the graph G into a new chains graph where each edge with label type RR is guaranteed to admit a range of repulsion as its beginning and end vertex. Hence, it will indicate the location of a repeller.

It will be shown in Corollary 3.2.62 that any edge with label type RR and same start and end vertex represents an R–chain. However:

Lemma 3.2.55. *For a tree pair (D, R, t) representing $\alpha \in V$ and its chains graph G, an edge with label type RR with distinct start and end vertices corresponds to an SRS–chain.*

Proof. Suppose that $(v, w, (s)R(u_1, \ldots, u_l)R)$ is an edge for some $v, w \in V_G$ such that $v \neq w$, some non-negative integer l, some word $s \in \{0,1\}^+$ and some neutral leaves $u_1, \ldots, u_l$ of the tree pair (D, R, t). We know by Lemma 3.2.22 that v and w cannot be neutral vertices of G. Hence both v and w are range vertices. Recall that by Lemma 3.2.24 that this edge corresponds to a non-periodic leaf chain $(vs, u_1, \ldots, u_l, w)$. We are interested in what type of leaves vs and w are. As v is a range vertex, vs is a leaf of a component of $D - R$, and hence it can be either a repeller or a source. But w, the last leaf in the leaf chain starting at vs, is not a proper ancestor of vs, hence vs must be a source. Now, as w is a range vertex, it is a root of a component of $D - R$, and hence it can be either a range of repulsion or a range of sourcing. But vs, the first leaf in the leaf chain ending at w, is not a proper descendant of w, hence w must be a range of sourcing.

As vs is a source and w is a range of sourcing, by Definition 3.1.27 the leaf chain $(vs, u_1, \ldots, u_l, w)$ corresponding to the edge

$$(v, w, (s)R(u_1, \ldots, u_l)R)$$

is an SRS–chain. □

As we know that SRS–chains cannot occur in revealing tree pairs, Algorithms 3.2.57 and 3.2.64 are aimed at eliminating these edges.

The penultimate algorithm presents an intermediate step, which eliminates *bottom vertices* (defined below) from the graph.

Definition 3.2.56 (Bottom vertex)**.** Consider a chains graph G which is an output of Algorithm 3.2.34 and which corresponds to the tree pair (D, R, t) representing element α of V. Let v be a range vertex of G. If each of its outgoing edges has label type RD, then we call v a *bottom vertex* of the graph G.

Note that the prerequisite for the two last algorithms is only Algorithm 3.2.34.

Algorithm 3.2.57. Consider a chains graph G which is an output of Algorithm 3.2.34 and which corresponds to the tree pair (D, R, t) representing element α of V.

1. Pick a bottom vertex v of the chains graph G. As it is a leaf of the range tree R, by Lemma 3.2.38 it has exactly one incoming edge, and this edge has label type RR. Say that this edge starts at a vertex w. In any case $(w, v, (\ldots)R(\ldots)R) \in E_G$. Note that as w is a start vertex of an edge of the label type RR, w is also a range vertex.

   Let the edge between w and v be given by

   $$(w, v, (s)R(u_1, \ldots, u_k)R)$$

   for some non–negative integer k, some neutral leaves $u_1, \ldots, u_k$ of the tree pair (D, R, t) and some word $s \in \{0, 1\}^+$.

   Consider each edge starting at v. By definition of bottom vertex, it needs to be of the form $(v, u, (s')R(u'_1, u'_2, \ldots, u'_{k'})D(s''))$ for some vertex $u \in V_G$, some non-negative integer k', some neutral leaves $u'_1, \ldots, u'_{k'}$ of the tree pair (D, R, t) and some words $s', s'' \in \{0, 1\}^+$. As v is a range vertex, at least two such edges exist by Lemma 3.2.11 and Lemma 3.2.20.

   Let us now perform the following process:

   (a) Replace each such edge

   $$(v, u, (s')R(u'_1, u'_2, \ldots, u'_{k'})D(s''))$$

with the edge

$$(w, u, (ss')R(u_1s', \ldots, u_ks', vs', u'_1, \ldots, u'_{k'})D(s'')).$$

(b) Delete the vertex v and the edge $(w, v, (s)R(u_1, \ldots, u_k)R)$.

2. Repeat until there are no more bottom vertices left.

Lemma 3.2.58. *Consider a chains graph which is an output of Algorithm 3.2.34 which we input into Algorithm 3.2.57. Then, the algorithm terminates with the chains graph G which admits no bottom vertices. Moreover, the resulting graph still represents the same element α of V.*

Proof. Note that with each cancellation of a bottom vertex, the vertex w might or might nor turn into a bottom vertex. However, the total number of vertices of the chains graph goes down by one in each iteration of the process, and hence the algorithm has to terminate. As termination is marked by the point when the chains graph admits no more bottom vertices, we have proven the first part of the claim.

For the second part, we will look more closely at what effect the edge replacements have on prefix substitution rules. Recall that we will be deleting the edge $e = (w, v, (s)R(u_1, \ldots, u_k)R)$ for distinct vertices v and w, where v is a bottom vertex. The prefix substitution rules given by this edge are:

$$(1) \begin{cases} (ws)\alpha = u_1 & \\ (u_l)\alpha = u_{l+1} & \forall 1 \leq l < k \\ (u_k)\alpha = v & \end{cases}$$

Let us consider what it would mean for the prefix substitution rules if without changing α instead of having one whole leaf v travel to the leaf ws under the action of negative powers of α, we split the leaf v further into smaller pieces, namely introduce a tree rooted at the leaf v of the tree R of the tree pair (D, R, t), at all neutral leaves $u_1, \ldots, u_k$ of both trees D and R, and at the leaf ws of the tree D.

An example of a tree which we could introduce could be the tree D_v rooted at the node v of the tree D. Then the prefix substitutions could be equally described by (1'):

$$(1') \begin{cases} \forall s' \in L_{D_v} & \\ (wss')\alpha = u_1 s' & \\ (u_l s')\alpha = u_{l+1} s' & \forall 1 \leq l < k \\ (u_k s')\alpha = vs' & \end{cases}$$

We could then cancel the edge e and represent these new prefix substitutions by a collection of following edges:

$$\{(w, vs', (ss')R(u_1 s', \ldots, u_k s')R) | s' \in L_{D_v}\}$$

We know that the label on each of the new edges has to be of type RR, because D_v is chosen depending on the origins of outgoing edges of v. Also, the operation does not impact the fact that w is still a range vertex of the graph. However, even after adjusting vertices, these edges would not form the chains graph corresponding to the new tree pair. This is because there would be vertices with an outgoing edge with first letter of the label being D and with an incoming edge with the second letter of the label being R. Algorithm 3.2.57 is inspired by the principle of concatenation of such neighbouring edges described in Remark 3.2.18. Let us analyse the effect of the suggested replacements of edges:

For each word $s' \in L_{D_v}$, the edge

$$(v, u, (s')R(u'_1, u'_2, \ldots, u'_{k'})D(s''))$$

is replaced by the edge

$$(w, u, (ss')R(u_1 s', \ldots, u_k s', vs', u'_1, \ldots, u'_{k'})D(s'')).$$

Let us analyse what it means in terms of prefix substitution rules. The initial edge encodes the following rules (2):

$$(2) \begin{cases} (vs')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \leq l < k' \\ (u'_{k'})\alpha = us'' & \end{cases}$$

The resulting edge encodes the following rules (3):

$$(3)\begin{cases} \left.\begin{array}{ll} (wss')\alpha = u_1 s' & \\ (u_l s')\alpha = u_{l+1} s' & \forall 1 \le l < k \\ (u_k s')\alpha = vs' & \end{array}\right\}(1') \\ \left.\begin{array}{ll} (vs')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \le l < k' \\ (u'_{k'})\alpha = us'' & \end{array}\right\}(2) \end{cases}$$

Notice that for given s' the rules (3) are a union of rules $(1')$ and (2). Hence, there has been no change to α in this procedure.

From this observation we can conclude that the resulting graph is indeed the chains graph of a new tree pair, which represents the same element α of V, as the prefix substitution rules remain the same.

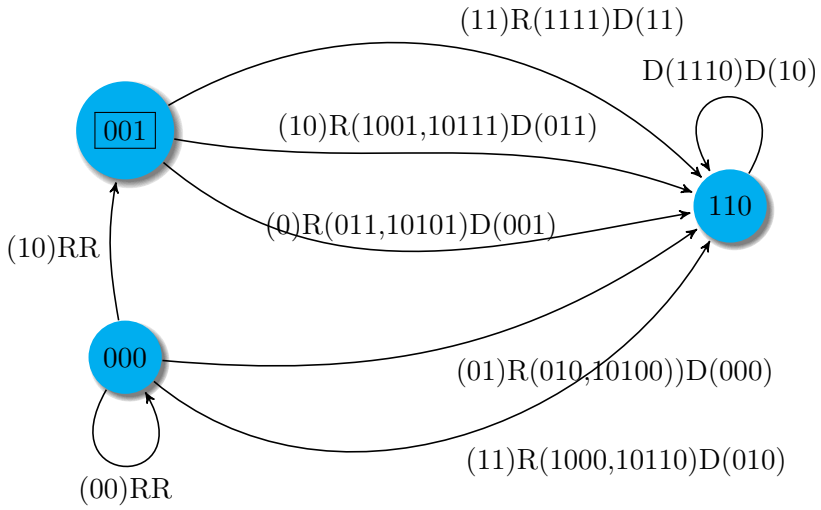
□

It is also important to realise that application of Algorithm 3.2.57 does not cancel the benefits of Algorithm 3.2.34:

Lemma 3.2.59. *Consider a chains graph G which is output by Algorithm 3.2.34 and Algorithm 3.2.57 applied one after another. Then G remains unchanged by a subsequent application of Algorithm 3.2.34, and so all the properties of a graph which is output by Algorithm 3.2.34 are preserved.*

Proof. The start vertex of each edge with label type DR of graph output by Algorithm 3.2.34 is also the end vertex of that edge. Algorithm 3.2.57 neither cancels nor introduces any edges with label type DR. Hence, if the graph G is an output of Algorithm 3.2.34 and Algorithm 3.2.57 applied one after another, it is still the case that beginning vertex of each edge with label type DR is also the end vertex of that edge. Therefore, the graph G remains unchanged under potential application of Algorithm 3.2.34. □

Example 3.2.60. Consider the chains graph G given in the picture below, which is an output of Algorithm 3.2.51 from Example 3.2.54:



The Chains Graph G

We notice that the graph G has no edges with label type DR (and in particular no edges with label type DR with distinct beginning and end vertices). Hence by Lemma 3.2.35 it remains unchanged by Algorithm 3.2.34. We can therefore input it into Algorithm 3.2.57.

In the picture we indicated the only bottom vertex of the graph G.

We now proceed with the application of Algorithm 3.2.57. Consider the vertex 001.

1. It has one incoming edge $(000, 001, (10)RR)$.

2. It has three outgoing edges, namely

$$(001, 110, (11)R(1111)D(11))$$
$$(001, 110, (0)R(011, 10101)D(001))$$
$$(001, 110, (10)R(1001, 10111)D(011)).$$

3. We replace the edge $(001, 110, (11)R(1111)D(11))$ with the edge

$$(000, 110, (1011)R(00111, 1111)D(11)).$$

4. We replace the edge $(001, 110, (0)R(011, 10101)D(001))$ with the

edge

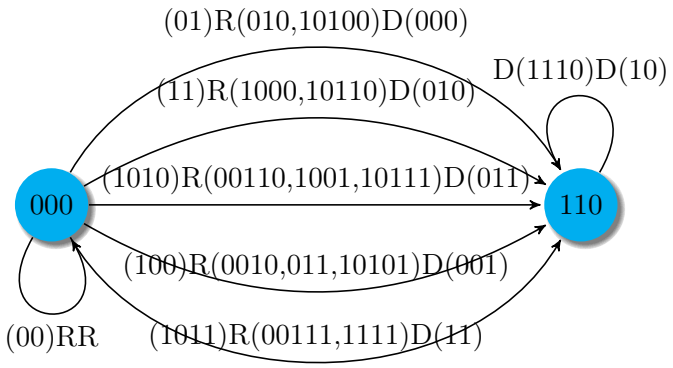
$$(000, 110, (100)R(0010, 011, 10101)D(001)).$$

5. We replace the edge $(001, 110, (10)R(1001, 10111)D(011))$ with the edge

$$(000, 110, (1010)R(00110, 1001, 10111)D(011)).$$

6. We delete the vertex 001 and the edge $(000, 001, (10)RR)$.

The new chains graph is given by the following picture:



The New Chains Graph

At this point the chains graph admits only one range vertex 000, but it has an outgoing edge with the label type RR, and hence it is not a bottom vertex. Hence, Algorithm 3.2.57 terminates.

We could interpret Algorithm 3.2.44 as prolonging appropriate edges with label type RD towards potential repeller(s). In the example above we detect precisely one repeller by the following lemma:

Lemma 3.2.61. *Consider a chains graph G corresponding to the tree pair (D, R, t) representing an element α of Thompson's group V. Consider a range vertex v of G and its unique incoming edge*

$$(w, v, (s)R(u_1, \ldots, u_k)R)$$

for some vertex w of G, some non-negative integer k, some neutral leaves $u_1, \ldots, u_k$ of the tree pair (D, R, t) and some word $s \in \{0, 1\}^*$. Then $v = w$ if and only if v is a range of repulsion leaf of the tree R and vs is a repeller leaf of the tree D.

Proof. Suppose that $v = w$. Consider the edge $(v, v, (s)R(u_1, \ldots, u_k)R)$ of the graph G. It indicates that α admits the following prefix replacements for all positive integers i such that $1 \leq i < k$:

$$\begin{aligned}(vs)\alpha &= u_1\\ (u_i)\alpha &= u_{i+1}\\ (u_k)\alpha &= v\end{aligned}$$

By Definition 3.1.7 and Definition 3.1.8, v is a range of repulsion leaf of the tree R and vs is a repeller leaf of the tree D.

Now, suppose that v is a range of repulsion leaf of R. Then there is a unique sequence of leaves $(v)\alpha^{-l}, (v)\alpha^{-l+1}, \ldots, (v)\alpha^{-1}, v$ of the tree pair (D, R, t) for some positive integer l, such that $(v)\alpha^{-l+1}, \ldots, (v)\alpha^{-1}$ are neutral leaves of the tree pair (D, R, t), $(v)\alpha^{-l}$ is a leaf of domain tree D but not range tree R, and $(v)\alpha^{-l}$ is a proper descendant of v. This is reflected in the graph G by existence of an edge $(v, v, (s')R((v)\alpha^{-l+1}, \ldots, (v)\alpha^{-1})R)$ for some non-empty word s' such that $vs' = (v)\alpha^{l}$. But v has a unique incoming edge, and so $w = v$, $k = l - 1$, for all positive integers i such that $1 \leq i < k$ we have $u_i = (v)\alpha^{i-k-1}$ and $s = s'$. Note that this implies that vs is a repeller. □

Corollary 3.2.62. *Consider a chains graph G corresponding to the tree pair (D, R, t) representing an element α of Thompson's group V. Suppose that there is an edge e of the label type RR starting and finishing at the same vertex. Then, the edge e corresponds to an A–chain of leaves.*

Proof. By Lemma 3.2.61, Lemma 3.2.24 and Definition 3.1.26. □

The last algorithm transforms a chains graph which is an output of Algorithm 3.2.34 and Algorithm 3.2.57 applied one after another into a new chains graph where each edge with label type RR is guaranteed to admit a range of repulsion as its beginning and end vertex, and hence locates a repeller. Before we present this algorithm, we will prove the following lemma:

Lemma 3.2.63. *Consider a chains graph G which is an output of Algorithm 3.2.34 and Algorithm 3.2.57 applied one after another. Then,*

each range vertex has a unique outgoing edge with label type RR. Moreover, each range vertex v_1 is a member of a unique set of range vertices $\{v_1, \ldots, v_j\}$ for some positive integer j, such that for all integers i such that $1 \leq i < j$ there is an edge $(v_{i+1}, v_i, (...)R(...)R) \in E_G$ and also $(v_1, v_j, (...)R(...)R) \in E_G$.

Proof. We will first show that there is a bijection between the set of range vertices and the set of edges with label type RR. Then we will show that given that no bottom vertices are allowed, the conclusions of the lemma must hold.

As by Lemma 3.2.59 the graph G remains unchanged by Algorithm 3.2.34, by Lemma 3.2.38 we know that each range vertex admits precisely one incoming edge, and the label type of this edge is RR. The end of an edge with label type RR cannot be a domain vertex, and by Lemma 3.2.21 it also cannot be a neutral vertex (as neutral vertices of chains graph are only at the beginnings and ends of edges of label type DR). Hence, the end vertex of an edge with label type RR is always a range vertex. Note also that end vertex of any edge is unique. Hence, there is exactly same number of range vertices as edges with label type RR in the graph G.

Now notice that by Lemma 3.2.58 none of the range vertices can be a bottom vertex and therefore by Lemma 3.2.11 and Lemma 3.2.20 each of range vertices must have at least one outgoing edge with label type RR. But each edge with label type RR has unique end vertex which is a range vertex. Hence as there is exactly same number of range vertices as edges with label type RR, by the pigeonhole principle, each range vertex must admit precisely one outgoing edge with label type RR.

Finally, let us consider a range vertex v_1. It has precisely one incoming edge, and the label type of this edge is RR. Hence we can build a unique sequence $(v_i)_{\mathbb{N}^+}$ such that the graph G admits an edge $(v_{i+1}, v_i, (\ldots)R(\ldots)R)$. However, as $|V_G| < \infty$, there will be repetitions of vertices in the sequence. For any given v_i which occurs in the sequence for the first time, namely $v_i \neq v_k$ for any positive integer k such that $k < i$, the vertex v_{i+1} can either be a previously unlisted vertex, or it can be v_1, as all the other listed vertices $v_2, v_3, \ldots, v_i$ already have their unique outgoing edge with label type RR identified. As $|V_G| < \infty$, we must at some point finish at the vertex v_1, say $v_{j+1} = v_1$ for some positive integer j. When we close the loop, all of the vertices involved in

the loop already have their unique incoming and outgoing edges of type RR identified, so they cannot be used to build any other sequences with starting vertices outside the set $\{v_1, v_2, \ldots, v_j\}$.

Therefore, we have proven the lemma. □

We can partition the range vertices into sets belonging to loops joined by edges with label type RR.

Now we are prepared to proceed with the algorithm after which we will have all the repellers which we need for a given α of V:

Algorithm 3.2.64. Consider a chains graph G which is an output of Algorithm 3.2.34 and Algorithm 3.2.57 applied one after another. Suppose that G corresponds to the tree pair (D, R, t) representing element α of V.

1. Pick a range vertex v such that its unique incoming edge with label type RR does not end at v. This vertex will remain in the graph.

   Identify the unique edge of label type RR coming into the vertex v, say $(u, v, (s)R(u_1, u_2, \ldots, u_k)R)$ for some vertex $u \in V_G$, some non–negative integer k, some neutral leaves $u_1, u_2, \ldots, u_k$ of the tree pair (D, R, t) and some word $s \in \{0, 1\}^+$.

   Identify the unique edge of label type RR coming into the vertex u, say $(w, u, (s')R(u'_1, u'_2, \ldots, u'_{k'})R)$ for some vertex $w \in V_G$, some non-negative integer k', some neutral leaves $u'_1, u'_2, \ldots, u'_{k'}$ of the tree pair (D, R, t) and some word $s' \in \{0, 1\}^+$.

   Then perform the following action:

   (a) Replace the edge $(u, v, (s)R(u_1, u_2, \ldots, u_k)R)$ with the edge

   $$(w, v, (s's)R(u'_1 s, u'_2 s, \ldots, u'_{k'} s, us, u_1, u_2, \ldots, u_k)R).$$

   Consider each other edge starting at u. By Lemma 3.2.63 it must be of the form $(u, u', (s'')R(u''_1, u''_2, \ldots, u''_{k''})D(s'''))$ for some vertex u', some non–negative integer k'', some neutral leaves $u''_1, \ldots, u''_{k''}$ of the tree pair (D, R, t) and some words $s'', s''' \in \{0, 1\}^+$.

(b) Replace each such edge with the edge

$$(w, u', (s's'')R(u'_1 s'', u'_2 s'', \dots, \\ \dots, u'_{k'} s'', us'', u''_1, u''_2, \dots, \\ \dots, u''_{k''})D(s'''))).$$

(c) Delete the vertex u and the edge

$$(w, u, (s')R(u'_1, u'_2, \dots, u'_{k'})R).$$

(d) Repeat the process until the unique edge with the label type RR finishing at the vertex v also starts at v.

2. Repeat Step 1) until all range vertices are at the same time beginning and end vertices of their unique incoming edges.

Lemma 3.2.65. *Consider a chains graph G which is an output of Algorithm 3.2.34, Algorithm 3.2.57 and Algorithm 3.2.64 applied one after another. Suppose that G corresponds to the tree pair (D, R, t) representing element α of V.*

Then the algorithm terminates with a new chains graph, with each of its range vertices corresponding to a range of repulsion leaf, and hence locating a repeller.

Moreover, each connected component of $D - R$ contains a repeller.

Also, the resulting graph still represents the same element α of V.

Proof. The algorithm terminates as each of its iterations reduces total number of vertices by 1. The algorithm is defined to terminate when each range vertex is at the same time beginning and end of its unique incoming edge. By Lemma 3.2.61 this means that each of the range vertices v of G corresponds to a range of repulsion leaf, and hence locates a repeller vs for some non–empty word s.

Moreover, connected components of $D-R$ are in bijection with range vertices of G, as each of the connected components of $D - R$ is rooted at a range vertex. By Lemma 3.2.61 vs is a repeller and it's located at a leaf of the connected component rooted at v. As this holds for each range vertex v, we conclude that each connected component of $D - R$ contains a repeller.

For the second part, we will look more closely at what effect the edge replacements have on prefix substitution rules. Recall that we will be deleting the edge $e = (w, u, (s')R(u'_1, u'_2, \ldots, u'_{k'})R)$ for distinct vertices v and w. The prefix substitution rules given by this edge are:

$$(1)\begin{cases} (ws')\alpha = u'_1 & \\ (u'_l)\alpha = u'_{l+1} & \forall 1 \leq l < k' \\ (u'_{k'})\alpha = u & \end{cases}$$

Let us consider what it would mean for the prefix substitution rules if without changing α instead of having one whole leaf u travel to the leaf ws' under the action of negative powers of α, we split the leaf u further into smaller pieces, introducing a tree rooted at the leaf u of the tree R of the tree pair (D, R, t), at all neutral leaves $u_1, \ldots, u_k$ of both trees D and R, and at the leaf ws' of the tree D.

An example of a tree which we could introduce could be the tree D_u rooted at the node u of the tree D. Then the prefix substitutions could be equally described by $(1')$:

$$(1')\begin{cases} \forall s'' \in L_{D_u} & \\ (ws's'')\alpha = u'_1 s'' & \\ (u'_l s'')\alpha = u'_{l+1} s'' & \forall 1 \leq l < k' \\ (u'_{k'} s'')\alpha = us'' & \end{cases}$$

We could then cancel the edge e and represent these new prefix substitutions by the collection of following edges:

$$\{(w, us'', (s's'')R(u_1 s'', \ldots, u_k s'')R) | s'' \in L_{D_u}\}$$

We know that the labels on each of the new edges have to be of the type RR, because D_u is chosen depending on the origins of outgoing edges of u. Also, w is still a range vertex of the new chains graph. However, even after adjusting vertices, these edges would not form the chains graph corresponding to the new tree pair. This is because there would be vertices with an outgoing edge with first letter of the label being D and with an incoming edge with the second letter of the label being R. Algorithm 3.2.64 is inspired by the principle of concatenation of such neighbouring edges described in Remark 3.2.18. Let us analyse the effect of the suggested replacements of edges:

1. For each word $s'' \in L_{D_u} \backslash \{s\}$, the edge

$$(u, u', (s'')R(u''_1, u''_2, \ldots, u''_{k''})D(s'''))$$

is replaced by the edge

$$(w, u', (s's'')R(u'_1 s'', u'_2 s'', \ldots, u'_{k'} s'', us'', u''_1, u''_2, \ldots, u''_{k''})D(s''')).$$

Let us analyse what it means in terms of prefix substitution rules. The initial edge encodes the following rules (2):

$$(2) \left\{ \begin{array}{rl} (us'')\alpha = u''_1 & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \leq l < k'' \\ (u''_{k''})\alpha = u's''' & \end{array} \right.$$

The resulting edge encodes the following rules (3):

$$(3) \left\{ \begin{array}{rl} (ws's'')\alpha = u'_1 s'' & \\ (u'_l s'')\alpha = u'_{l+1} s'' & \forall 1 \leq l < k' \\ (u'_{k'} s'')\alpha = us'' & \end{array} \right\} (1') \qquad \left\{ \begin{array}{rl} (us'')\alpha = u''_1 & \\ (u''_l)\alpha = u''_{l+1} & \forall 1 \leq l < k'' \\ (u''_{k''})\alpha = u's''' & \end{array} \right\} (2)$$

Notice that for given s'' the rules (3) are a union of rules $(1')$ and (2). Hence, there has been no change to α in this procedure.

2. The edge $(u, v, (s)R(u_1, u_2, \ldots, u_k)R)$ is replaced with the edge $(w, v, (s's)R(u'_1 s, u'_2 s, \ldots, u'_{k'} s, us, u_1, u_2, \ldots, u_k)R)$. Let us analyse what it means in terms of prefix substitution rules. The initial edge encodes the following rules (4):

$$(4) \left\{ \begin{array}{rl} (us)\alpha = u_1 & \\ (u_l)\alpha = u_{l+1} & \forall 1 \leq l < k \\ (u_k)\alpha = v & \end{array} \right.$$

The resulting edge encodes the following rules (5):

$$(5)\begin{cases} \left.\begin{array}{rl} (ws's)\alpha = u'_1 s & \\ (u'_l s)\alpha = u'_{l+1} s & \forall 1 \leq l < k' \\ (u'_{k'} s)\alpha = us & \end{array}\right\}(1') \\ \left.\begin{array}{rl} (us)\alpha = u_1 & \\ (u_l)\alpha = u_{l+1} & \forall 1 \leq l < k \\ (u_k)\alpha = v & \end{array}\right\}(4) \end{cases}$$

Notice that for $s'' = s$ the rules (5) are a union of rules $(1')$ and (4). Hence, there has been no change to α in this procedure.

By this observation we can conclude that the resulting graph is indeed a chains graph of a new tree pair, which represents the same element α of V as the prefix substitution rules remain the same.

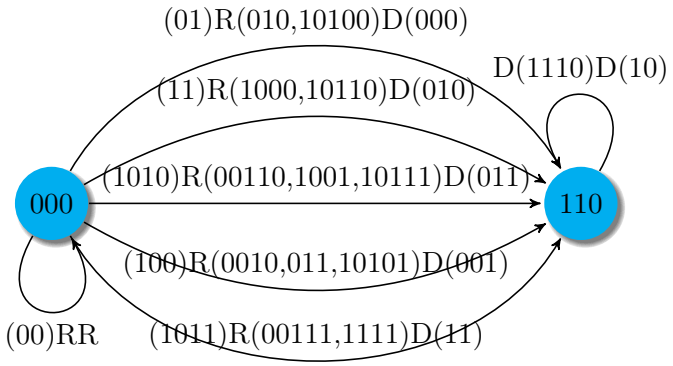
□

It is also important to realise that application of Algorithm 3.2.64 does not cancel the benefits of Algorithm 3.2.34 and Algorithm 3.2.57:

Lemma 3.2.66. *Consider a chains graph G which is an output of Algorithm 3.2.34, Algorithm 3.2.57 and Algorithm 3.2.64 applied one after another. Then G remains unchanged by a subsequent applications of Algorithm 3.2.34 and Algorithm 3.2.57, and so it keeps all the properties of a graph which is an output of Algorithm 3.2.34 and of a graph which is an output of Algorithm 3.2.57.*

Proof. The start vertex of each edge with label type DR of an output graph of Algorithm 3.2.34 is also the end vertex of that edge. Each of Algorithm 3.2.57 and Algorithm 3.2.64 neither cancels nor introduces any edges with label type DR. Hence, if the graph G is an output of Algorithm 3.2.34, Algorithm 3.2.57 and Algorithm 3.2.64 applied one after another, it is still the case that beginning vertex of each edge with label type DR is also the end vertex of that edge. Therefore, the graph G remains unchanged under potential application of Algorithm 3.2.34. Moreover, by Lemma 3.2.61 Lemma 3.2.65, each range vertex of the graph G is a beginning vertex of an edge of the label type RR, and hence it is not a bottom vertex. Hence, if the graph G is an output of Algorithm 3.2.34, Algorithm 3.2.57 and Algorithm 3.2.64 applied one after another, it is still the case that it admits no bottom vertices. Therefore, the graph G remains unchanged under potential application of Algorithm 3.2.57. □

Example 3.2.67. Consider the chains graph G given in the picture below:



The Chains Graph G

Notice that G is a final chains graph from Example 3.2.60, and hence an output of Algorithm 3.2.34 and Algorithm 3.2.57 applied one after another. We can therefore input it into Algorithm 3.2.64 in order to create all possible repellers.

We notice that there is only one range vertex 000, and it has the edge $(000, 000, (00)RR)$ beginning and ending at it. Hence, the algorithm terminates here without any action. We can deduce that 000 is the unique range of repulsion leaf of the tree R, and 00000 is the unique repeller leaf of the domain tree D.

Conclusion and Interpretation

We are now ready to prove two Theorems which were the motivation for this section - that for a given element α of Thompson's group V, there always exists a revealing tree pair representing it, and that for any given tree pair (D, R, t) representing α, we found a series of algorithms which transform (D, R, t) into a revealing tree pair.

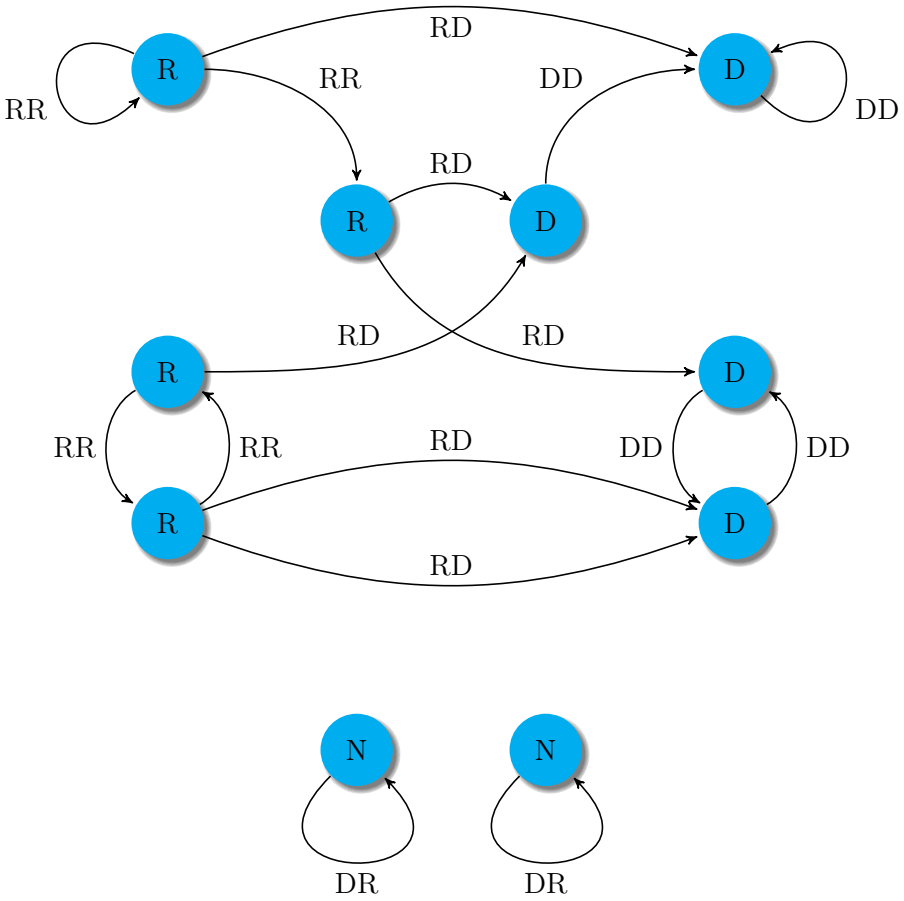
We start from a lemma which describes the relationships between some of the constituent algorithms.

Lemma 3.2.68. *Consider a chains graph G which is an output of Algorithm 3.2.34 and which corresponds to the tree pair (D, R, t) representing element α of V.*

Then, application of Algorithm 3.2.44 and subsequently Algorithm 3.2.51 commutes with application of Algorithm 3.2.57 and subsequently Algorithm 3.2.64.

This in particular implies that all properties of chains graphs resulting from application of any of these four algorithms to G are valid for the output graph, as long as Algorithm 3.2.44 is applied before Algorithm 3.2.51 and as long as Algorithm 3.2.57 is applied before Algorithm 3.2.64.

Proof. Consider the following exemplary sketch of the behaviour of the graph G:



Sketch of the Graph G

If we consider possible behaviours occuring in the graph G, we will notice that we can partition vertices of G into three categories: range vertices (denoted by R), domain vertices (denoted by D) and neutral vertices (denoted by N).

The collection of neutral vertices is not connected to any of the range or domain vertices, or even to each other as a matter of fact. By Lemma 3.2.21 each neutral vertex admits an edge with the label type DR which starts and finishes at it. By Lemma 3.2.35 the graph G admits no other edges with the label type DR. None of Algorithm 3.2.44, Algorithm 3.2.51, Algorithm 3.2.57 or Algorithm 3.2.64 has any effect on any neutral vertices or any edges of label type DR. Hence the effect of these algorithms on these parts of the graph commute.

The collection of range vertices admits all edges with the label type RR among each other, as no domain vertex can have an edge with the label type RR pointing in or out of it. The collection of domain vertices admits all edges with the label type DD among each other, as no range vertex can have an edge with the label type DD pointing in or out of it. The only remaining part of the graph which needs describing is the collection of edges with label type RD. Each such edge starts at some range vertex and finishes at some domain vertex.

Consider areas of impact of Algorithm 3.2.44 and subsequently Algorithm 3.2.51. Each step of each of these algorithms deals with and changes domain vertices, edges with label type DD, and prolongs edges with label type RD towards the end. Notice that at no point beginning vertex or any of the neutral vertices of an edge with label type RD has been amended.

Similarly, consider areas of impact of Algorithm 3.2.57 and subsequently Algorithm 3.2.64. Each step of each of these algorithms deals with and changes range vertices, edges with label type RR, and prolongs edges with label type RD towards the beginning. Notice that at no point end vertex or any of the neutral vertices of an edge with label type RD has been amended.

In conclusion, as the following pairs: Algorithm 3.2.44 and Algorithm 3.2.51; Algorithm 3.2.57 and Algorithm 3.2.64, act non–trivially on distinct and disjoint parts of the graph G, their effects on the graph G commute.

This proves the conclusions of this lemma. □

With the help of the lemma above we are finally ready to prove:

Theorem 3.2.69. *Consider a chains graph G which is an output of Algorithm 3.2.34 and subsequently Algorithm 3.2.44 and subsequently Algorithm 3.2.51; Algorithm 3.2.57 and subsequently Algorithm 3.2.64, in any*

order of application of these two pairs. Graph G corresponds to the tree pair (D, R, t) representing element α of V.

Then, the tree pair (D, R, t) is a revealing tree pair.

Proof. Recall Definition 3.1.17 of a revealing tree pair and Lemma 3.2.68. By Lemma 3.2.52 each connected component of $R - D$ admits an attractor, and by Lemma 3.2.65 each connected component of $D - R$ admits a repeller, and hence (D, R, t) is a revealing tree pair. □

Hence, we have the following theorem as a corollary:

Theorem 3.2.70. *For any element $\alpha \in V$, there always exists a revealing pair (D, R, t) representing it.*

Proof. By Theorem 3.2.69. □

A natural question to ask is why we decided to present five distinct mandatory algorithms and one optional for finding a revealing pair for given α.

The reason for presenting the optional algorithm is its potential to reduce the amount of data which needs to be handled in the problem. Yet, we did not want to create the impression that it is indispensable, especially given that in some cases it might involve steps which would need to be reversed by later algorithms.

Algorithm 3.2.34 needs to be performed as the first one among the five essential algorithms. We feel that it deserves to exist in its own right due to strong properties possessed by its output graph. This algorithm uncovers all periodic orbits of leaves and hence can be used separately e.g. for this purpose only, for deciding whether α is torsion or whether α has any torsion at all.

There is benefit in realising that the search for attractors and repellers can be performed separately, and for that reason we separated Algorithms 3.2.44 and 3.2.51 from Algorithms 3.2.57 and 3.2.64. However, what were our reasons for splitting the search for attractors into eliminating top vertices and picking the attractors, and similarly splitting the search for repellers into eliminating bottom vertices and picking the repellers? The main reason is that in each of Algorithms 3.2.51 and 3.2.64 we need to make an explicit choice of which vertex remains in the graph and so where the repeller/attractor will occur. Hence we wanted to leave the option of not applying these two algorithms. Hence, we define the following:

Definition 3.2.71 (Balanced Pair)**.** Consider a chains graph G which is an output of Algorithm 3.2.28, subsequently Algorithm 3.2.34, and subsequently Algorithms 3.2.44 and 3.2.57 in any order of application of these last two. Graph G corresponds to a uniquely determined tree pair (D, R, t) representing element α of V. The tree pair (D, R, t) is called a *balanced (tree) pair.*

Due to the deterministic nature of all algorithms involved in the definition above, we believe that there is a unique balanced tree pair for each $\alpha \in V$.

Finally, for a better understanding of how the resulting chains graph of the resulting revealing pair links with leaf chains and previous way in which we defined revealing pairs, we present this summary. At the end of the process for obtaining a revealing pair, we end up with four types of edges. They correspond to 4 types of chains:

Corollary 3.2.72. *The resulting graph of the entire sequence of described algorithms has each edge corresponding to a leaf chain as follows:*

1. *A vertex with an edge with label DR starting and finishing at itself corresponds to a P–chain of leaves of the resulting tree pair.*

2. *A vertex with an edge with label DD starting and finishing at itself corresponds to an A–chain of leaves of the resulting tree pair.*

3. *A vertex with an edge with label RR starting and finishing at itself corresponds to an R–chain of leaves of the resulting tree pair.*

4. *An edge with label RD corresponds to an SS–chain of leaves of the resulting tree pair.*

Proof. Each of the above holds by:

1. Corollary 3.2.25,

2. Corollary 3.2.49,

3. Corollary 3.2.62,

4. Lemma 3.2.27.

□

3.3 Rollings

We know from Section 3.2 how to obtain a revealing pair for an element α of Thompson's group V, given any tree pair representing it. However for any given α there are many revealing pairs corresponding to it. In this section we will use leaf chains to rephrase the work of Salazar–Díaz [28] on how to find all revealing pairs for α when initially given only one of its revealing tree pairs. We will use this information to find all *important points*, of an element α of V, which we will introduce in Section 3.4.

A *rolling* of a given revealing pair (D, R, t) representing an element α of V is another revealing tree pair representing the same element α, which has been obtained from the tree pair (D, R, t) using certain allowable operations. We can move from any revealing pair to any other revealing pair using rolling and inverse rolling operations, as proved by Salazar–Díaz in [28].

Types of Rollings

We distinguish between three types of rollings – type E (standing for elementary), type I and type II. We will describe algorithms used to produce each of them. For clarity, we will give examples of each of the rolling types.

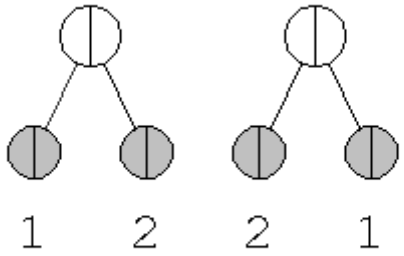
Algorithm 3.3.1 (Rolling Type E (Elementary))**.** Consider a revealing pair (D, R, t) representing an element α of V.

1. Pick any SS–chain or P–chain, associated with this tree pair, given by $(\lambda\alpha^i)_{i=0}^k$ for some leaf $\lambda \in L_D$ and some non–negative integer k.
2. Let T be any binary rooted tree.
3. For every leaf $\lambda\alpha^i$ in the chosen chain:
   (a) If $\lambda\alpha^i \in L_D$, in order to obtain the new tree D' attach a copy of the tree T to D by identifying the root of T to the leaf $\lambda\alpha^i$ of D.
   (b) If $\lambda\alpha^i \in L_R$, in order to obtain the new tree R' attach a copy of the tree T to R by identifying the root of T to the leaf $\lambda\alpha^i$ of R.
4. Change the tuple t to obtain a new tuple t' such that the tree pair (D', R', t') represents α. To achieve this, apply extended version of

Algorithm 2.3.9 corresponding to all augmentations introduced to the initial tree pair.

Definition 3.3.2 (Rolling Type E)**.** Suppose that the revealing pair (D', R', t') representing $\alpha \in V$ has been obtained from a revealing pair (D, R, t) by using Algorithm 3.3.1, possibly numerous times. Then we call (D', R', t') a *rolling of type E* from (D, R, t).

Example 3.3.3 (Rolling Type E)**.** Let (D, R, t) be the following tree pair representing $\alpha \in V$:



We notice that the leaves of this tree pair form a single P–chain given by $(0, 1)$, as $0 \overset{\alpha}{\mapsto} 1 \overset{\alpha}{\mapsto} 0$. Hence this tree pair has no repellers or attractors. Also, both $D - R$ and $R - D$ are empty, and so the tree pair (D, R, t) is revealing.

Thus we can apply Algorithm 3.3.1 to its P–chain $(0, 1)$ to obtain a new tree pair (D', R', t') which will be a rolling of type E from (D, R, t):

1. Let T be a single caret.

2. As both 0 and 1 are in L_D, to create D' we attach a copy of the tree T to D by identifying the root of T to the leaf 0, and another copy of the tree T by identifying the root of T to the leaf 1.

3. As both 0 and 1 are in L_R, to create R' we attach a copy of the tree T to R by identifying the root of T to the leaf 0, and another copy of the tree T by identifying the root of T to the leaf 1.

4. The new leaves of the tree D' are given by: $L_{D'} = \{00, 01, 10, 11\}$. We label them with numbers from 1 to 4 respectively.

5. None of the leaves of D' is a leaf of D. Thus:

(a) The newly assigned number of the leaf 00 is 1. Hence we assign number 1 to the leaf $(00)\alpha = (0)\alpha 0 = 10$ of R'.

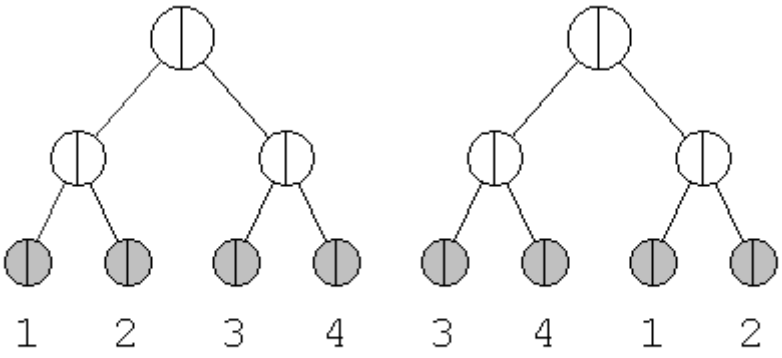
(b) The newly assigned number of the leaf 01 is 2. Hence we assign number 2 to the leaf $(01)\alpha = (0)\alpha 1 = 11$ of R'.

(c) The newly assigned number of the leaf 10 is 3. Hence we assign number 3 to the leaf $(10)\alpha = (1)\alpha 0 = 00$ of R'.

(d) The newly assigned number of the leaf 11 is 4. Hence we assign number 4 to the leaf $(11)\alpha = (1)\alpha 1 = 01$ of R'.

(e) In order to get t', we read the numbers of the tree R' in order from left to right: $t' = (3, 4, 1, 2)$.

Hence, the new tree pair (D', R', t') is given by:



Algorithm 3.3.4 (Rolling Type I)**.** Consider a revealing pair (D, R, t) representing an element α of V.

1. Pick any R–chain or A–chain associated with this tree pair given by $(\lambda\alpha^i)_{i=0}^k$ for some leaf $\lambda \in L_D$ and some non-negative integer k.

2. (a) If $(\lambda\alpha^i)_{i=0}^k$ is an R–chain, define the tree C to be the component of $D - R$ rooted at $\lambda\alpha^k$. Define Γ to be the word $a_1 \dots a_l$ for some positive integer l and $a_1, \dots a_l \in \{0, 1\}$, such that $\lambda = \lambda\alpha^k\Gamma$.

   (b) If $(\lambda\alpha^i)_{i=0}^k$ is an A–chain, define the tree C to be the component of $R - D$ rooted at λ. Define Γ to be the word $a_1 \dots a_l$ for some positive integer l and $a_1, \dots a_l \in \{0, 1\}$, such that $\lambda\Gamma = \lambda\alpha^k$.

3. We will now pick a tree T as follows:

   (a) Pick an integer j such that $1 \leq j \leq l$.

   (b) Split the tree C into two components T and T' where contains the root of C and T' is rooted at the address $a_1 a_2 \dots a_j$.

4. For every leaf $\lambda\alpha^i$ in the chosen chain:

   (a) If $\lambda\alpha^i \in L_D$, in order to obtain the new tree D' attach a copy of the tree T to D by identifying the root of T to the leaf $\lambda\alpha^i$ of D.

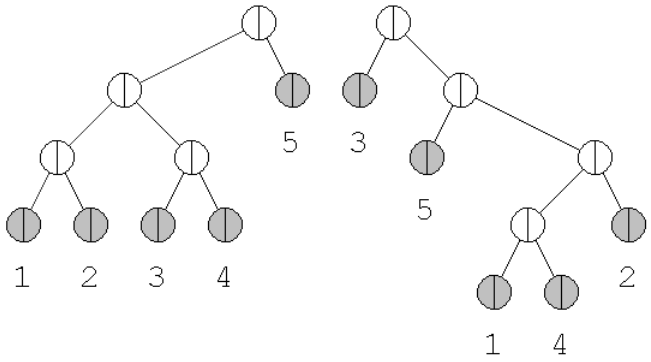
   (b) If $\lambda\alpha^i \in L_R$, in order to obtain the new tree R' attach a copy of the tree T to R by identifying the root of T to the leaf $\lambda\alpha^i$ of R.

5. Change the tuple t to obtain a new tuple t' such that the tree pair (D', R', t') represents α. To achieve this, apply extended version of Algorithm 2.3.9 corresponding to all augmentations introduced to the initial tree pair.

The word Γ for a given repeller or attractor will be of great importance in Section 3.4. We will define it to be a *spine* in Definition 3.4.2 and 3.4.11.

Definition 3.3.5 (Rolling Type I)**.** Suppose that the revealing pair (D', R', t') representing $\alpha \in V$ has been obtained from a revealing pair (D, R, t) by using Algorithm 3.3.4, possibly numerous times. Then we call (D', R', t') a *rolling of type I* from (D, R, t).

Example 3.3.6 (Rolling Type I)**.** Let (D, R, t) be the following tree pair representing $\alpha \in V$:



First we identify the leaf chains of the tree pair (D, R, t):

1. $000 \overset{\alpha}{\longmapsto} 1100$.
2. $001 \overset{\alpha}{\longmapsto} 111$.
3. $010 \overset{\alpha}{\longmapsto} 0$.
4. $011 \overset{\alpha}{\longmapsto} 1101$.
5. $1 \overset{\alpha}{\longmapsto} 10$.

We recognise 3. to be an R–chain, corresponding to the only component of $D - R$. We also recognise 5. to be an A–chain, corresponding to the only component of $R - D$. Hence the tree pair (D, R, t) is revealing and we can apply Algorithm 3.3.4 to it. For that, let us pick the R–chain $(010, 0)$ to obtain a new tree pair (D', R', t') which will be a rolling of type I from (D, R, t):

1. The word Γ is given by 10 and the tree C is given by the component of $D - R$ rooted at the node 0.
2. We pick $j = 1$ and so we cut the tree C at its node 1. The tree T is the component which contains the root of C.
3. We attach a copy of T to D by identifying the leaf 010 of D to the root of T to form D'. We also attach a copy of T to R by identifying the leaf 0 of R to the root of T to form R'.

4. The leaves of D' in order from left to right are as follows: $000, 001, 01000, 01001, 0101, 011, 1$. We label them from 1 to 7 respectively.

5. (a) The newly assigned number of the leaf 000 is 1. Hence we assign number 1 to the leaf $(000)\alpha = 1100$ of R'.

   (b) The newly assigned number of the leaf 001 is 2. Hence we assign number 2 to the leaf $(001)\alpha = 111$ of R'.

   (c) The newly assigned number of the leaf 01000 is 3. Hence we assign number 3 to the leaf $(01000)\alpha = (010)\alpha 00 = 000$ of R'.

   (d) The newly assigned number of the leaf 01001 is 4. Hence we assign number 4 to the leaf $(01001)\alpha = (010)\alpha 01 = 001$ of R'.

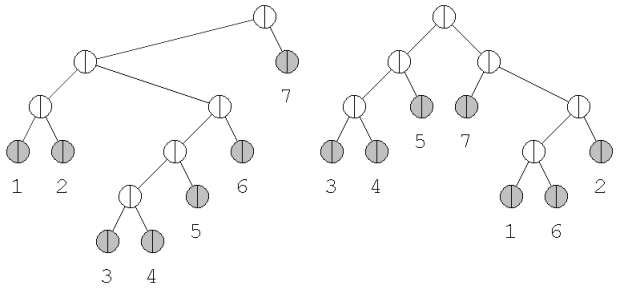
   (e) The newly assigned number of the leaf 0101 is 5. Hence we assign number 5 to the leaf $(0101)\alpha = (010)\alpha 1 = 01$ of R'.

   (f) The newly assigned number of the leaf 011 is 6. Hence we assign number 6 to the leaf $(011)\alpha = 1101$ of R'.

   (g) The newly assigned number of the leaf 1 is 7. Hence we assign number 7 to the leaf $(1)\alpha = 10$ of R'.

6. In order to get t', we read the numbers of the tree R' in order from left to right. As the leaves of R' occur in the following order: $000, 001, 01, 10,$
   $1100, 1101, 111$, we deduce that $t' = (3, 4, 5, 7, 1, 6, 2)$.

The resulting tree pair (D', R', t') is given by:



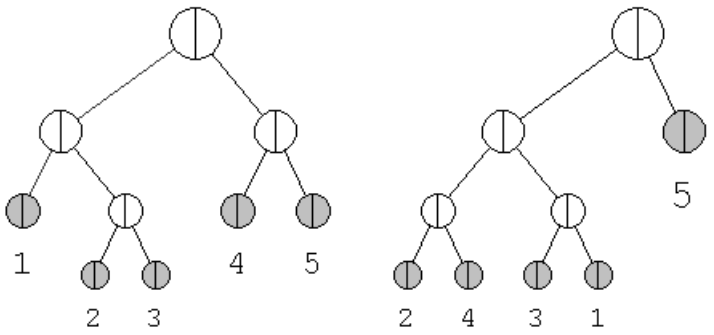
Algorithm 3.3.7 (Rolling Type II)**.** Consider a revealing pair (D, R, t) representing an element α of V.

- Pick any R–chain (or A–chain respectively) associated with this tree pair given by $(\lambda\alpha^i)_{i=0}^k$ for some leaf $\lambda \in L_D$ and some non–negative integer k.

- Let T be the component of $D - R$ (or $R - D$ respectively) rooted at the leaf $\lambda\alpha^k$ (or λ respectively).
- To create D', attach a copy of the tree T to D by identifying the root of T to the leaf $\lambda\alpha^{k-1}$ (or λ respectively).
- To create R', attach a copy of the tree T to R by identifying the root of T to the leaf $\lambda\alpha^k$ (or $\lambda\alpha$ respectively).
- Change the tuple t to obtain a new tuple t' such that the tree pair (D', R', t') represents α. To achieve this, apply Algorithm 2.3.9 to the initial tree pair.

Definition 3.3.8 (Rolling Type II)**.** Suppose that the revealing pair (D', R', t') representing $\alpha \in V$ has been obtained from a revealing pair (D, R, t) by using Algorithm 3.3.7, possibly numerous times. Then we call (D', R', t') a *rolling of type II* from (D, R, t).

Example 3.3.9 (Rolling Type II)**.** Let (D, R, t) be the following tree pair representing $\alpha \in V$:



We identify the leaf chains of the tree pair (D, R, t):

1. $00 \overset{\alpha}{\longmapsto} 011 \overset{\alpha}{\longmapsto} 010 \overset{\alpha}{\longmapsto} 000$.
2. $10 \overset{\alpha}{\longmapsto} 001$.
3. $11 \overset{\alpha}{\longmapsto} 1$.

We recognise 3. to be an R–chain corresponding to the only component of $D - R$. We also recognise 1. to be an A–chain corresponding to

the only component of $R - D$. Hence the tree pair (D, R, t) is revealing and we can apply Algorithm 3.3.7 to it. For that, let us pick the A–chain $(00, 011, 010, 000)$ to obtain a new tree pair (D', R', t') which will be of rolling type II from (D, R, t):

1. T is given by the only component of $R - D$, which is given by a single caret.

2. To create D' we attach a copy of T to D by identifying the leaf 00 of D to the root of T. To create R', we attach a copy of T to R by identifying the leaf 011 of R to the root of T.

3. The leaves of D' in order from left to right are as follows:

$$000, 001, 010, 011, 10, 11.$$

   We label them from 1 to 6 respectively.

4. (a) The newly assigned number of the leaf 000 is 1. Hence we assign number 1 to the leaf $(000)\alpha = (00)\alpha 0 = 0110$ of R'.

   (b) The newly assigned number of the leaf 001 is 2. Hence we assign number 2 to the leaf $(001)\alpha = (00)\alpha 1 = 0111$ of R'.

   (c) The newly assigned number of the leaf 010 is 3. Hence we assign number 3 to the leaf $(010)\alpha = 000$ of R'.

   (d) The newly assigned number of the leaf 011 is 4. Hence we assign number 4 to the leaf $(011)\alpha = 010$ of R'.

   (e) The newly assigned number of the leaf 10 is 5. Hence we assign number 5 to the leaf $(10)\alpha = 001$ of R'.

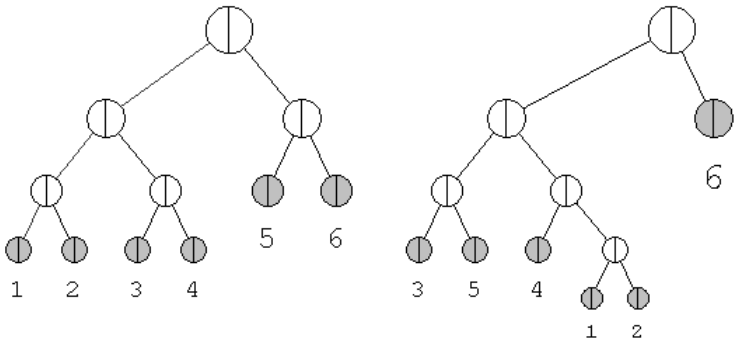
   (f) The newly assigned number of the leaf 11 is 6. Hence we assign number 6 to the leaf $(11)\alpha = 1$ of R'.

5. In order to get t', we read the numbers of the tree R' in order from left to right. As the leaves of R' occur in the following order:

$$000, 001, 010, 0110, 0111, 1$$

   we deduce that $t' = (3, 5, 4, 1, 2, 6)$.

The resulting tree pair is given by:



Theorem 3.3.10. *For an element α of Thompson's group V, we can transition from any revealing pair for α to any other revealing pair for α using rollings and inverse rollings described in Algorithms 3.3.1, 3.3.4 and 3.3.7.*

Proof. By Salazar–Díaz in [28]. □

3.4 Important Points of an Element of V

The term *important points* was first used in this context by Bleak and Salazar–Díaz in [8], the reason being that these points were of key importance for their consideration of free products in V. They will prove to be indispensable in Chapter 4 on centralisers in V, and of central focus in Chapter 5 on further properties of free products. They are described in this last section of this chapter in hope of creating a fresh, lasting impression suggested by recency effect.

In this section we will proceed to give a formal definition and explore basic properties of important points. Informally, they should be considered as a special kind of points from $\mathfrak{C}$ which lie on finite orbits under the action of a given element α of V, while all other points from small neighbourhood around them lie on infinite orbits. Moreover, the neighbourhood around them either expands or contracts, and depending on which of these two is occurring, we discriminate between *repelling* and *attracting* important points of α. This property of being isolated from other points lying on finite orbits discriminates important points from

those points of $\mathfrak{C}$ which underlie periodic neutral leaves of a revealing tree pair representing α.

Lemma 3.4.1. *Consider an element α of the Thompson's group V, and a revealing tree pair (D, R, t) representing it. Consider a repeller λ from L_D and its proper ancestor $(\lambda)\alpha^s$ from L_R for some positive integer s. Say that $(\lambda)\alpha^s$ has address γ_s and λ has address $\gamma_0 = \gamma_s\Gamma$, where γ_s, γ_0 and Γ are finite strings of numbers from the set $\{0, 1\}$. Then, $p_{\gamma_0} = \gamma_0(\Gamma)^\infty$ is the unique point of $\mathfrak{C}$ with the prefix γ_s which is fixed by the map α^s.*

Proof. As α^s induces a bijection on $\mathfrak{C}$ and the image of the set of all points with prefix $\gamma_0 = \gamma_s\Gamma$ is the set of all points with prefix γ_s, the only possible points fixed by α^s with prefix γ_s must in fact have prefix $\gamma_s\Gamma$. As α^s is such that it substitutes prefix $\gamma_s\Gamma$ with γ_s, then the point $\gamma_s(\Gamma)^\infty$ is the only point of $\mathfrak{C}$ with prefix $\gamma_s\Gamma$ which is mapped to itself by the action of α^s. Indeed, $\gamma_s(\Gamma)^\infty = \gamma_s(\Gamma(\Gamma)^\infty) = (\gamma_s\Gamma)(\Gamma)^\infty = \gamma_0(\Gamma)^\infty = ((\gamma_s)\alpha)(\Gamma)^\infty = (\gamma_s(\Gamma)^\infty)\alpha$. □

Definition 3.4.2 (Spine of a Repeller)**.** Consider an element α of Thompson's group V, and a revealing tree pair (D, R, t) representing it. Consider a repeller λ from L_D with and its ancestor $\lambda\alpha^s$ from L_R for some positive integer s. Say that $\lambda\alpha^s$ has address γ_s and λ has address $\gamma_0 = \gamma_s\Gamma$, where γ_s, γ_0 and Γ are finite strings of numbers from the set $\{0, 1\}$. Then, we call Γ the *spine* of the repeller λ.

Remark 3.4.3. Consider a revealing pair (D, R, t) of an element α of V and (D', R', t') a rolling of type I from it on a given R–chain with repeller $\lambda \in L_D$ (see Algorithm 3.3.4). Say that $\lambda\alpha^s$ is a proper ancestor of λ for some positive integer s. Say that λ has address γ_0 and $\lambda\alpha^s$ has address γ_s. Then $\gamma_s\Gamma = \gamma_0$ for a word $\Gamma = a_1a_2 \dots a_l$ which is the spine of repeller λ, where l is a positive integer and $a_1, a_2, \dots, a_l \in \{0, 1\}$. Then for the choice of integer j such that $1 \leq j \leq l$, determined by the rolling (D', R', t'), we define $\omega = a_1a_2 \dots a_j$ and $\omega' = a_{j+1} \dots a_l$. Then our new repeller $\lambda\omega$ of (D', R', t') has address $\gamma_0\omega$. Notice that $\gamma_0\omega = \gamma_s\omega\omega'\omega$. Notice also that $((\gamma_s\omega)\omega'\omega)\alpha^s = (\gamma_s\Gamma\omega)\alpha^s = (\gamma_s\Gamma)\alpha^s\omega = \gamma_s\omega$ so the spine of the new repeller $\lambda\omega$ is given by $\Gamma' = \omega'\omega = a_{j+1} \dots a_la_1a_2 \dots a_j$, which is a cyclic permutation of the letters in the word Γ.

Lemma 3.4.4. *Consider an element α of the Thompson's group V, and a revealing tree pair (D, R, t) representing it. Consider a repeller λ from*

5. $10 \overset{\alpha}{\longmapsto} 101$: A–chain.

6. $110 \overset{\alpha}{\longmapsto} 1100$: A–chain.

7. $111 \overset{\alpha}{\longmapsto} 1111$: A–chain.

We recognise 2. as an R–chain corresponding to the only component of $D - R$ rooted at 00. We also recognise 5., 6. and 7. to be A–chains corresponding to connected components of $R - D$ rooted at 10, 110 and 111 respectively. Hence, the tree pair (D, R, t) is revealing, and we can proceed to find all its repelling points.

The R–chain 2. consists of tree leaves $0001, 011, 00$ and so its repeller has spine 01. Thus, there are two corresponding repelling points given by

$$\mathcal{R}_\alpha = \{00(01)^\infty, 011(01)^\infty\}$$

Remark 3.4.7 ($\mathcal{R}_\alpha$ is finite). Note that each repelling point of α corresponds to a distinct leaf of the tree D, and as the set L_D is finite, the set $\mathcal{R}_\alpha$ must also be finite. Moreover, as the effect on α on $\mathfrak{C}$ is independent of the choice of a tree pair for it, $\mathcal{R}_\alpha$ is also independent of the choice of the tree pair representing α.

Definition 3.4.8 (Fixed Repelling Point of α)**.** Consider an element α of the Thompson's group V, and a revealing tree pair (D, R, t) representing it. If p is a repelling point of α and α fixes p, namely $(p)\alpha = p$, then p is a special kind of a repelling point called a *fixed repelling point of* α.

Note that a given revealing pair of an element α from V admits one repeller per orbit of repelling points under the action of α. The repeller highlights the position of one of the repelling points of α. The following lemma helps us recognise how to highlight other repelling points by choosing a different revealing pair:

Lemma 3.4.9. *Let (D, R, t) be a revealing tree pair corresponding to α of V. Say that $(\lambda\alpha^i)_{i=0}^s$ is an R–chain of (D, R, t) for some $\lambda \in L_D$ and some positive integer s. Suppose that for each i such that $0 \leq i \leq s$, the address of $\lambda\alpha^i$ is given by γ_i. Also $\gamma_0 = \gamma_s\Gamma$, where Γ is the spine of the repeller λ. Consider the repelling point $\gamma_i(\Gamma)^\infty$ which underlies the leaf $\lambda\alpha^i$, for $i \in \{0, \ldots, s-1\}$. In particular, consider $\gamma_0(\Gamma)^\infty$ which underlies the repeller λ.*

Now consider a single rolling (D', R', t') of type II from (D, R, t) on this R–chain. Then the leaf $\lambda\alpha^{s-1}\Gamma$ is a repeller of (D', R', t') overlying the repelling point $\gamma_{s-1}(\Gamma)^\infty$.

Proof. The tree pair (D, R, t) is such that for each i such that $1 \leq i \leq s-1$, $\lambda\alpha^i$ is a neutral leaf. Also, λ is a leaf of D and $\lambda\alpha^s$ is a leaf of R. Let T be the tree given by the component of $D - R$ rooted at $\lambda\alpha^s$. By Algorithm 3.3.7 the tree D' is the tree D expanded by attaching a copy of T by identifying the leaf $\lambda\alpha^{s-1}$ with the root of T. Also, the tree R' is the tree R expanded by attaching a copy of T by identifying the leaf $\lambda\alpha^s$ with the root of T. Hence, $\lambda, \lambda\alpha, \ldots, \lambda\alpha^{s-2}$ are neutral leaves of the tree pair (D', R', t'), $\lambda\alpha^{s-1}\Gamma$ is a leaf of the tree D' and $\lambda\alpha^{s-1}$ is a leaf of the tree R'. Notice that $((\gamma_0)\alpha^{s-1}\Gamma)\alpha = (\gamma_{s-1}\Gamma)\alpha = (\gamma_{s-1})\alpha\Gamma = \gamma_s\Gamma = \gamma_0$. Therefore, $(\lambda\alpha^{s-1}\Gamma, \lambda, \lambda\alpha, \ldots, \lambda\alpha^{s-1})$ is an R–chain of (D', R', t'). This implies that $\lambda\alpha^{s-1}\Gamma$ is a repeller, and the important point which it is overlying is $\gamma_{s-1}(\Gamma)^\infty$. □

After having defied and analysed properties of *repelling points of* α, we need to mention *attracting points of* α. Informally, attracting points of α can be understood as repelling points of α^{-1}, and hence their properties will be very similar:

Lemma 3.4.10. *Consider an element α of the Thompson's group V, and a revealing tree pair (D, R, t) representing it. Consider an attractor $\lambda\alpha^k$ from L_R and its ancestor λ from L_D for some positive integer k. Say that $\lambda\alpha^k$ has address γ_k and λ has address γ_0 such that $\gamma_0\Gamma = \gamma_k$, where γ_k, γ_0 and Γ are finite strings of numbers from the set $\{0, 1\}$. Then, $p_{\gamma_0} = \gamma_0(\Gamma)^\infty$ is the unique point of $\mathfrak{C}$ with the prefix γ_0 which is fixed by the map α^k.*

Proof. As α^k induces a bijection on $\mathfrak{C}$ and the image of the set of all points with prefix γ_0 is the set of all points with prefix $\gamma_0\Gamma$, the only possible points fixed by α^k with prefix γ_0 must in fact have prefix $\gamma_0\Gamma$. As α^k is such that it substitutes prefix γ_0 with $\gamma_0\Gamma$, then the point $\gamma_0(\Gamma)^\infty$ is the only point of $\mathfrak{C}$ with prefix $\gamma_0\Gamma$ which is mapped to itself by the action of α^k. Indeed, $\gamma_0(\Gamma)^\infty = \gamma_0(\Gamma(\Gamma)^\infty) = (\gamma_0\Gamma)(\Gamma)^\infty = \gamma_k(\Gamma)^\infty = ((\gamma_0)\alpha)(\Gamma)^\infty = (\gamma_0(\Gamma)^\infty)\alpha$. □

Definition 3.4.11 (Spine of an Attractor)**.** Consider an element α of Thompson's group V, and a revealing tree pair (D, R, t) representing it.

Consider an attractor $(\lambda)\alpha^k$ from L_R with and its proper ancestor λ from L_D for some positive integer k. Say that $(\lambda)\alpha^k$ has address γ_k and λ has address γ_0 such that $\gamma_k = \gamma_0\Gamma$, where γ_k, γ_0 and Γ are finite strings of numbers from the set $\{0, 1\}$. Then, we call Γ the *spine* of the attractor $(\lambda)\alpha^k$.

Remark 3.4.12. Consider a revealing pair (D, R, t) of an element α of V and (D', R', t') a rolling of type I from it on a given A–chain with attractor $\lambda\alpha^k \in L_R$ (see Algorithm 3.3.4). Say that λ is a proper ancestor of $\lambda\alpha^k$ for some positive integer k. Say that λ has address γ_0 and $\lambda\alpha^k$ has address γ_k. Then $\gamma_k = \gamma_0\Gamma$ for a word $\Gamma = a_1a_2 \dots a_l$ which is the spine of attractor $\lambda\alpha^k$, where l is a positive integer and $a_1, a_2, \dots, a_l \in \{0, 1\}$. Then for a choice of integer j such that $1 \leq j \leq l$, determined by the rolling (D', R', t'), we define $\omega = a_1a_2 \dots a_j$ and $\omega' = a_{j+1} \dots a_l$. Then our new attractor $\lambda\alpha^k\omega$ of (D', R', t') has address $\gamma_k\omega$. Notice that $\gamma_k\omega = \gamma_0\omega\omega'\omega$. Notice also that $((\gamma_0\omega))\alpha^k = (\gamma_0)\alpha^k\omega = \gamma_k\omega = (\gamma_0\omega)\omega'\omega$, so the spine of the new attractor $\lambda\alpha^k\omega$ is given by $\Gamma' = \omega'\omega = a_{j+1} \dots a_la_1a_2 \dots a_j$, which is a cyclic permutation of the letters in the word Γ.

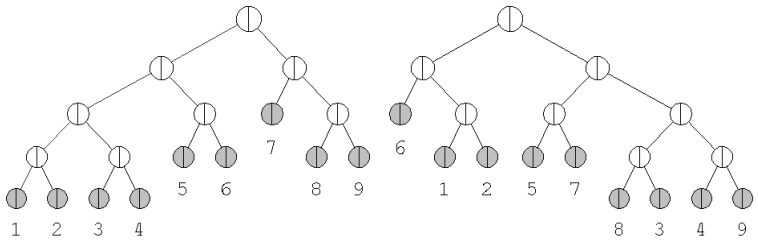
Lemma 3.4.13. *Consider an element α of the Thompson's group V, and a revealing tree pair (D, R, t) representing it. Consider an attractor $\lambda\alpha^k$ from L_R and the sequence of leaves $\lambda\alpha^i$ for $0 \leq i \leq k$ such that λ is a proper ancestor of $\lambda\alpha^k$. For $0 \leq i \leq k$ let γ_i be the address of the leaf $\lambda\alpha^i$, and let $\gamma_0\Gamma = \gamma_k$ for $\Gamma \in \{0, 1\}^+$. Then for each $0 \leq i \leq k$, $p_{\gamma_i} = \gamma_i(\Gamma)^\infty$ is the unique point of $\mathfrak{C}$ with the prefix γ_i which is fixed by the map α^k.*

Proof. Notice that for all $0 \leq i \leq k$ we have $(\gamma_i)\alpha^k = (\gamma_0)\alpha^i\alpha^k = (\gamma_k)\alpha^i = (\gamma_0\Gamma)\alpha^i = (\gamma_0)\alpha^i\Gamma = \gamma_i\Gamma$. Hence we deduce that an application of α^k has an effect of substituting prefix $\gamma_i\Gamma$ in place of γ_i. Hence, the lemma holds as a corollary of Lemma 3.4.10. □

Definition 3.4.14 (Periodic Attracting Point of α)**.** Consider an element α of the Thompson's group V, and a revealing tree pair (D, R, t) representing it. Consider an attractor $\lambda\alpha^k$ from L_D and the sequence of leaves $\lambda\alpha^i$ for $0 \leq i \leq k$ such that $\lambda\alpha^k$ is a proper descendant of λ. For $0 \leq i \leq k$ let γ_i to be the address of the leaf $\lambda\alpha^i$ and let $\gamma_0\Gamma = \gamma_k$. Then, for each $0 \leq i < k$ we will call $\gamma_i(\Gamma)^\infty$ a *periodic attracting point of* α, or simply an *attracting point of* α. Also, let the set of all attracting points of α be called $\mathcal{A}_\alpha$.

Note that the word *periodic* is meant to emphasise the fact that attracting points of α lie on finite orbits under the action of α.

Example 3.4.15 (Attracting Points)**.** Consider the revealing tree pair (D, R, t) given by the same picture as in Example 3.4.6:



Recall its leaf chains:

1. $0000 \overset{\alpha}{\longmapsto} 010 \overset{\alpha}{\longmapsto} 100$: SS–chain.
2. $0001 \overset{\alpha}{\longmapsto} 011 \overset{\alpha}{\longmapsto} 00$: R–chain.
3. $0010 \overset{\alpha}{\longmapsto} 1101$: SS–chain.
4. $0011 \overset{\alpha}{\longmapsto} 1110$: SS–chain.
5. $10 \overset{\alpha}{\longmapsto} 101$: A–chain.
6. $110 \overset{\alpha}{\longmapsto} 1100$: A–chain.
7. $111 \overset{\alpha}{\longmapsto} 1111$: A–chain.

The A–chain 5. consists of two leaves $10, 101$ and so its attractor has spine 1. Thus there is one corresponding attracting point given by $10(1)^\infty$.

The A–chain 6. consists of two leaves $110, 1100$ and so its attractor has spine 0. Thus there is one corresponding attracting point given by $110(0)^\infty$.

The A–chain 7. consists of two leaves $111, 1111$ and so its attractor has spine 1. Thus there is one corresponding attracting point given by $111(1)^\infty$.

$$\mathcal{A}_\alpha = \{10(1)^\infty, 110(0)^\infty, 111(1)^\infty\}$$

Remark 3.4.16 ($\mathcal{A}_\alpha$ is finite). Note that each attracting point of α corresponds to a distinct leaf of the tree R, and as the set L_R is finite, the set $\mathcal{A}_\alpha$ must also be finite. Moreover, as the effect on α on $\mathfrak{C}$ is independent of the choice of a tree pair for it, $\mathcal{A}_\alpha$ is also independent of the choice of the tree pair representing α.

Definition 3.4.17 (Fixed Attracting Point of α)**.** Consider an element α of the Thompson's group V, and a revealing tree pair (D, R, t) representing it. If p is an attracting point of α and α fixes p, namely $(p)\alpha = p$, then p is a special kind of an attracting point called a *fixed attracting point of* α.

Note that a given revealing pair of an element α from V admits one attractor per orbit of attracting points under the action of α. The attractor highlights the position of one of the attracting points of α. The following lemma helps us recognise how to highlight other attracting points by choosing a different revealing pair:

Lemma 3.4.18. *Let (D, R, t) be a revealing tree pair corresponding to α of V. Say that $(\lambda\alpha^i)_{i=0}^k$ is an A–chain of (D, R, t) for some $\lambda \in L_D$ and some positive integer k. Suppose that for each i such that $0 \leq i \leq k$, the address of $\lambda\alpha^i$ is given by γ_i. Also $\gamma_0\Gamma = \gamma_k$, where Γ is the spine of the attractor $\lambda\alpha^k$. Consider the attracting point $\gamma_i(\Gamma)^\infty$ which underlies the leaf $\lambda\alpha^i$, for $i \in \{0, \ldots, s-1\}$. In particular, consider $\gamma_k(\Gamma)^\infty$ which underlies the attractor λ.*

Now consider a single rolling (D', R', t') of type II from (D, R, t) on this A–chain. Then the leaf $\lambda\alpha\Gamma$ is an attractor of (D', R', t') overlying the repelling point $\gamma_1(\Gamma)^\infty$.

Proof. The tree pair (D, R, t) is such that for each i such that $1 \leq i \leq s-1$, $\lambda\alpha^i$ is a neutral leaf. Also, λ is a leaf of D and $\lambda\alpha^k$ is a leaf of R. Let T be the tree given by the component of $R - D$ rooted at λ. By Algorithm 3.3.7 the tree D' is the tree D expanded by attaching a copy of T by identifying the leaf λ with the root of T. Also, the tree R' is the tree R expanded by attaching a copy of T by identifying the leaf $\lambda\alpha$ with the root of T. Hence, $\lambda\alpha^2, \ldots \lambda\alpha^k, \lambda\Gamma$ are neutral leaves of the tree pair (D', R', t'), $\lambda\alpha$ is a leaf of the tree D' and $\lambda\alpha\Gamma$ is a leaf of the tree R'.

Notice that $((\gamma_0)\alpha^k)\alpha = (\gamma_k)\alpha = (\gamma_0\Gamma)\alpha = (\gamma_0)\alpha\Gamma = \gamma_1\Gamma$. Therefore, $(\lambda\alpha, \ldots, \lambda\alpha^k, \lambda\Gamma, \lambda\alpha\Gamma)$ is an A–chain of (D', R', t'). This implies that

$\lambda\alpha\Gamma$ is an attractor, and the important point which it is overlying is $\gamma_1(\Gamma)^\infty$. □

Definition 3.4.19 (Important Points of α)**.** Consider an element α of Thompson's group V. Let the *set of important points of* α denoted by $\mathcal{I}_\alpha$ be defined as $\mathcal{I}_\alpha = \mathcal{R}_\alpha \sqcup \mathcal{A}_\alpha$.

Remark 3.4.20. Note that the set of points underlying neutral periodic leaves of a revealing pair and important points of α is equal to the set of points of $\mathfrak{C}$ lying on finite orbits under the action of α. This has been implicitly proven in Lemma 3.2.39. Moreover, there is a universal bound on the sizes of all finite orbits in $\mathfrak{C}$ under the action of α. This is because of the following: there is a finite number of attractors and repellers for a revealing tree pair for α. Each of the repellers and attractors is in one–to–one correspondence with a finite orbit of important points under α. Hence the set $\mathcal{I}_\alpha$ is finite. The remaining points on finite orbits lie precisely under periodic neutral leaves of a revealing pair for α. Each of these leaves has a finite orbit under the action of α, and there are also finitely many of them. Hence we can put a universal bound on the size of the finite collection of finite orbits Let us say that it is n where n is a positive integer. Therefore, we can also find a positive integer, for intstance $m = n!$, such that α^m admits only orbits of size 1 or ∞.

Chapter 4

Centralisers

In this chapter we intend to familiarise the reader with the structure of the centraliser group of an element α of Thompson's group V. The main theorem of Bleak *et al.* [5] states the structure of the centraliser of an element of V. In this thesis we provide a converse to this theorem, by proving that each of the predicted structures is realisable. Hence we obtain a complete classification of centralisers in V. We first give an overview on general structure of $C_V(\alpha)$ as derived in [5]. Then we discuss in detail the theory which led directly to obtaining our results.

In [5], the centraliser is proven to split as a direct product of two groups: one acting on the points from $\mathfrak{C}$ lying in the closure of infinite orbits under the element's action, and the other acting on the remaining points from $\mathfrak{C}$, all of which lie on finite orbits. While the general structures of both groups are established in [5], it remained unclear whether all possible realisations of the structure are achievable. (See questions (2) and (3) in [5].)

In our work we use results from [5] to derive an algorithm for constructing an element $\alpha \in V$ with strictly prescribed centraliser. We achieve that through embedding sufficient symmetry into its flow graph object, so that the flow graph automorphisms realises the Cayley action for any given finite group of our choice. These automorphisms give rise to elements centralising α. Thus we conclude that anything allowed by the classification of Bleak *et al.* is in fact realisable.

We make explicit use of revealing pairs and important points, as discussed in Chapter 3, to construct flow graph objects as introduced by Bleak *et al.* in [5].

4.1 Statement of Results

Definition 4.1.1. Define $\mathcal{C}$ to be the class of all groups isomorphic to a group of the following form:

$$\left(\prod_{i=1}^{s} M_{m_i} \rtimes V\right) \times \left(\prod_{j=1}^{t} ((A_j \wr_{\psi_j} \mathbb{Z}) \wr P_{q_j})\right)$$

for positive integers s, m_i (s.t. $m_1 < m_2 < ... < m_s$) and q_j, groups M_{m_i}, A_j, and P_{q_j}, and $\psi_j \in Aut(A_j)$, such that $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, t\}$, and such that:

- for each i, $M_{m_i} = \text{Maps}(\mathfrak{C}, \mathbb{Z}_{m_i})$, where $\text{Maps}(\mathfrak{C}, \mathbb{Z}_{m_i})$ is the group of continuous maps from $\mathfrak{C}$ to $\mathbb{Z}_{m_i}$ under pointwise addition, and where $\mathbb{Z}_{m_i}$ is the cyclic group $\mathbb{Z}/m_i\mathbb{Z}$ under the discrete topology;
- for each j, A_j is a finite group;
- for each j, P_{q_j} is the full symmetric group on q_j elements.

Our main result of this chapter combined with the work by Bleak *et al.* in [5] is the following theorem:

Theorem 4.1.2. *Let $\mathcal{C}$ be the class of groups from Definition 4.1.1. Then a group G belongs to $\mathcal{C}$ if and only if there is an element α in Thompson's group V such that G is isomorphic to $C_V(\alpha)$, the centraliser of α in V.*

We will give a very brief overview of what the class $\mathcal{C}$ corresponds to dynamically. For more details, see [5].

Remark 4.1.3 (Flow Graph Properties). In order to give a dynamical interpretation of Theorem 4.1.2, we need to briefly explain basic facts about flow graphs. We will include a more detailed description of flow graphs in Section 4.2.

A flow graph will in many ways resemble a chains graph of a revealing tree pair for an $\alpha \in V$. A flow graph of an element $\alpha \in V$ is a disjoint union of connected graph components of two possible types: periodic and non-periodic. The periodic type represents the points from $\mathfrak{C}$ on finite orbits under the action of α (excluding its important points $\mathcal{I}_\alpha$), which are precisely the ones underlying leaves of P–chains of a revealing pair for α. The non–periodic type corresponds to the closure of points from $\mathfrak{C}$ lying on infinite orbits under the action of α, which are precisely

the points underlying leaves of R–chains, A–chains and SS–chains of a revealing pair for α.

Now, in Definition 4.1.1, if the given group is $C_V(\alpha)$, then:

- The number s denotes number of distinct lengths of the P–chains of a revealing pair for α.
- Each group of the form $M_{m_i} \rtimes V$ corresponds to a subgroup of the centraliser acting on those points from $\mathfrak{C}$ which are underlying the leaves of P–chains of length m_i of a revealing pair for α.
- Each group of the form $A_j \rtimes_{\psi_j} \mathbb{Z}$ represents a subgroup of the centraliser corresponding to points in $\mathfrak{C}$ supported under a single component of the flow graph of α, which set is the closure of the support of a set of infinite orbits.
- The number t indicates number of equivalence classes of non–periodic flow graph components.
- Each q_j is the size of the j^{th} equivalence class of non–periodic flow graph components.
- The action of the group P_{q_j} on $\mathfrak{C}$ induces a permutation on q_j isomorphic non-periodic flow graph components.

In [5] it has been proven that if a group is realisable as a centraliser of an element $\alpha \in V_n$, then it takes the form described in Theorem 4.1.2. It remained an open question which finite groups A_j (Question (2) by Bleak *et al.* in [5]), and which automorphisms $\psi_j \in Aut(A_j)$ producing the semidirect action (Question (3) by Bleak *et al.* in [5]) could occur. Explicit examples of abelian A_j were known and indeed the original question was whether A_j needs to be abelian. An obstacle to understanding which groups were achievable is that, as A_j becomes more complex, the elements α which induce A_j for this part of their centraliser are rapidly growing in complexity as well (as measured for instance by the complexity of their minimal representative tree pairs, the size of the flow graph etc.). No examples of non-trivial ψ_j were known. The new work presented in this chapter gives rise to the following two theorems:

Theorem 4.1.4. *For any finite group A, there exists an $\alpha \in V$ such that:*

1. *a revealing pair for α has a connected non–periodic (CNP) flow graph;*

2. *A is isomorphic to the torsion subgroup of α's centraliser.*

Theorem 4.1.5. *For any finite group A and any automorphism ψ of A, there exists an $\alpha_\psi \in V$ such that:*

1. *a revealing pair for $\alpha_\psi^{|\psi|}$ has a connected flow graph, which is non–periodic;*

2. *$\alpha_\psi^{|\psi|}$ has centraliser isomorphic to $A \rtimes_\psi \mathbb{Z}$ with α_ψ generating the subgroup $\mathbb{Z}$.*

Without developing the appropriate theory it would be a strong claim to say that a particular example does not have more elements in its centraliser. Therefore, examples for both of the theorems above are exhibited in the last section of this chapter.

4.2 Literature Review

We will recall the results from [5] which we will be using in this chapter. We will also rephrase and expand on them wherever we find it beneficial.

Flow Graphs

The flow graph of an element $\alpha \in V$ is a structure modelling dynamical properties of the element α's effect on Cantor space. It provides benefits complimentary to these provided by tree pairs. It was first introduced by Bleak *et al.* in [5] in order to provide better intuition for analysing the centraliser of α. A flow graph is a directed labelled graph constructed from a revealing pair for α. It is similar to the chains graph of this revealing tree pair, but it emphasises different information carried by the element α.

Here is how to construct the flow graph of a revealing tree pair:

Algorithm 4.2.1. Consider a revealing pair (D, R, t) for α.

1. List all the leaf chains for the revealing pair (D, R, t).

2. For each P–chain we draw a vertex for each of the neutral leaves of this chain and we label each vertex with the address of this neutral

leaf. We draw an edge from one of these vertices to another if a single iteration of α takes the first leaf to the second, provided that these two leaves are not the same.

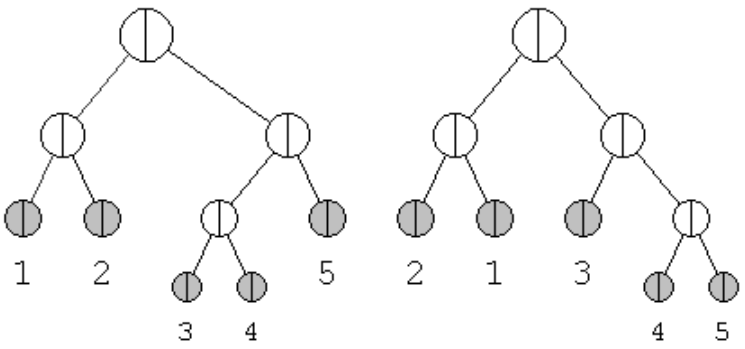
3. For each R–chain we draw a vertex. If the R–chain is longer than two leaves, draw a directed edge starting and ending at this vertex. Let the R–chain be given by $(\lambda_i)_{i=0}^{k}$, and $\lambda_0 = \lambda_k\Gamma$ for the spine Γ. Label the vertex with a sequence of addresses of corresponding repelling points $\lambda_0(\Gamma)^\infty - \lambda_1(\Gamma)^\infty - \ldots - \lambda_{k-1}(\Gamma)^\infty$.

4. For each A–chain we draw a vertex. If the A–chain is longer than two leaves, draw a directed edge starting and ending at this vertex. Let the A–chain be given by $(\lambda'_i)_{i=0}^{k'}$, and $\lambda'_0\Gamma' = \lambda'_{k'}$ for the spine Γ'. Label the vertex with a sequence of addresses of corresponding attracting points $\lambda'_0(\Gamma')^\infty - \lambda'_1(\Gamma')^\infty - \ldots - \lambda'_{k'-1}(\Gamma')^\infty$.

5. For each SS–chain we draw a directed edge. The first vertex of the SS–chain belongs to a component of $D - R$ which contains a single repeller. Let v be the vertex (added in 3. above) corresponding to the R–chain of this repeller. The last vertex w of the SS–chain belongs to a component of $R - D$ which contains a single attractor. Let w be the vertex corresponding to the A–chain of this attractor. Add an edge beginning at v and ending at w. Let the SS–chain be given by $(\lambda''_i)_{i=0}^{k''}$ for some positive integer k. We label the edge with the sequence of addresses $\lambda''_0 - \lambda''_1 - \ldots - \lambda''_{k''}$.

We will display a basic flow graph in the example below:

Example 4.2.2. Let $\alpha \in V$ be represented by the following tree pair (D, R, t):



The leaves of the tree D which are not leaves of the tree R are 100, 101 and 11. Hence we can identify the leaf chains of this tree pair.

1. $100 \xrightarrow{\alpha} 10$. As 10 is a prefix for 100, this is an R–chain.

2. $101 \xrightarrow{\alpha} 110$. As 101 is a leaf of connected component of $D - R$ and 110 is a leaf of connected component of $R - D$, this is an SS–chain.

3. $11 \xrightarrow{\alpha} 111$. As 11 is a prefix for 111, this is an A–chain.

4. The neutral leaves which remained unused in the leaf chains above are 00 and 01. In fact, we have that $00 \xrightarrow{\alpha} 01 \xrightarrow{\alpha} 00$. Hence, $(00, 01)$ is a P–chain.

Thus we conclude that the tree pair is revealing as there is one repeller corresponding to the only connected component of $D - R$ and there is one attractor corresponding to the only connected component of $R - D$. Hence we proceed as follows to construct the flow graph for the tree pair (D, R, t):

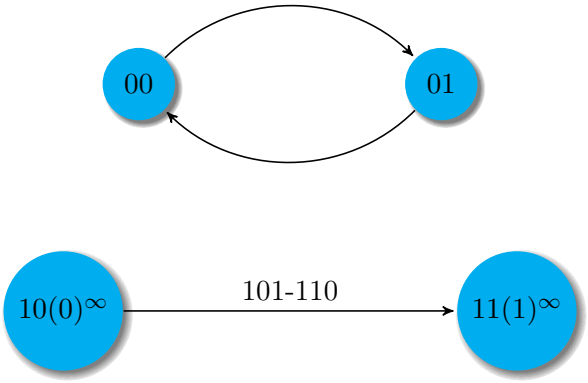
1. For the P–chain $(00, 01)$ we draw two vertices and we label them with addresses 00 and 01 respectively. Then we draw a directed edge from the vertex 00 to the vertex 01, and a directed edge from the vertex 01 to the vertex 00.

2. For the R–chain $(110, 10)$ we draw a vertex. We label the vertex with the address of the underlying repelling point, namely $10(0)^\infty$.

3. For the A–chain $(11, 111)$ we draw a vertex. We label the vertex with the address of the underlying attracting point, namely $11(1)^\infty$.

4. For the SS–chain $(101, 110)$ we draw an edge starting at the vertex $10(0)^\infty$ and ending at the vertex $11(1)^\infty$. We label the edge with the sequence of addresses $101 - 110$.

The resultant flow graph is as follows:



The Flow Graph of the Tree Pair (D, R, t)

Remark 4.2.3. As originally defined, the flow graph is constructed using a revealing tree pair representing α, but in this chapter we relax this condition and allow flow graphs arising from some specific types of tree pairs, called *balanced tree pairs*, which do not need to be revealing. In particular these will be tree pairs where chains graphs remain unchanged by Algorithms 3.2.34, 3.2.44 and 3.2.57, but not necessarily by Algorithms 3.2.51 and 3.2.64. We use them implicitly in Construction 4.3.13. The reason for doing so is to obtain a higher level of symmetry for these objects and their corresponding tree pairs.

The Centraliser's Action on Important Points

We will now prove that for $\alpha \in V$ the centraliser $C_V(\alpha)$ of α acts on the set $\mathcal{R}_\alpha$ of repelling points of α and also on the set $\mathcal{A}_\alpha$ of attracting points of α. This fact emphasises the very restrictive nature of elements centralising α, and at the same time provides us with a very powerful tool for the subsequent analysis of $C_V(\alpha)$.

Notice that all the statements in this subsection are true even if we assume that α has no important points, in which case the action of $C_V(\alpha)$ on $\mathcal{I}_\alpha$ is trivial.

We start with the following lemma:

Lemma 4.2.4. *Let α be an element of Thompson's group V. Consider any $r \in \mathcal{R}_\alpha$ and $c \in C_V(\alpha)$. Then $(r)c \in \mathcal{R}_\alpha$. Similarly, if $a \in \mathcal{A}_\alpha$, then $(a)c \in \mathcal{A}_\alpha$.*

Proof. Let k be the smallest positive integer k such that $(r)\alpha^k = r$. Now, pick a prefix λ of r such that the image $(\lambda)c$ is a single prefix, and

$(\lambda\Gamma)\alpha^k = \lambda$ for Γ the spine of r. Thus $r = (\lambda)\Gamma^\infty$. This is possible as r is a repelling point of α. Then, as c commutes with the action of α^k on Cantor space, we have $(\lambda)c = ((\lambda\Gamma)\alpha^k)c = (\lambda\Gamma)c\alpha^k = ((\lambda)c\Gamma)\alpha^k)$. Hence, the prefix $(\lambda)c\Gamma$ is substituted by $(\lambda)c$ under the action of α^k. Hence, α^k fixes the point $(\lambda)c\Gamma^\infty$ and $(\lambda)c\Gamma^\infty$ is a repelling point of α. Note that $(r)c = ((\lambda)\Gamma^\infty)c = (\lambda)c\Gamma^\infty$, which proves the claim.

A very similar argument proves that if $a \in \mathcal{A}_\alpha$, then $(a)c \in \mathcal{A}_\alpha$. Let k be the smallest positive integer k such that $(a)\alpha^k = a$. Now, pick a prefix λ of a such that $(\lambda)\alpha^k = \lambda\Gamma$ for Γ a spine of a and $(\lambda)c$ is given by a single prefix. Thus $a = (\lambda)\Gamma^\infty$. This is possible as a is an attracting point of α. Then, as c commutes with an action of α^k on Cantor space, we have $(\lambda)c\Gamma = (\lambda\Gamma)c = ((\lambda)\alpha^k)c = (\lambda)c\alpha^k = ((\lambda)c)\alpha^k)$. Hence, the prefix $(\lambda)c$ is substituted by $(\lambda)c\Gamma$ under the action of α^k and thus, α^k fixes the point $(\lambda)c\Gamma^\infty$. Note $(\lambda)c\Gamma^\infty$ is an attracting point of α and that $(a)c = ((\lambda)\Gamma^\infty)c = (\lambda)c\Gamma^\infty$, which proves the claim. □

From the lemma above, we can conclude the following useful facts:

Corollary 4.2.5. *Consider an element $\alpha \in V$. The group $C_V(\alpha)$ acts on the set $\mathcal{R}_\alpha$ and on the set $\mathcal{A}_\alpha$.*

Proof. We already know that as $c \in V$, it acts on Cantor space. As c is a bijection on Cantor space and $\mathcal{R}_\alpha$ is finite, by Lemma 4.2.4 we have $(\mathcal{R}_\alpha)c = \mathcal{R}_\alpha$, so $C_V(\alpha)$ acts on the set $\mathcal{R}_\alpha$. Similarly, by Lemma 4.2.4 we have $(\mathcal{A}_\alpha)c = \mathcal{A}_\alpha$ and so $C_V(\alpha)$ acts on the set $\mathcal{A}_\alpha$. □

Finally, the corollary below gives us another way to view the action of $C_V(\alpha)$ on $\mathcal{R}_\alpha$. We include it here because later in this section we will explicitly work with the map q which it mentions:

Corollary 4.2.6. *For a given $\alpha \in V$, there is a homomorphism q such that:*

$$q : C_V(\alpha) \longrightarrow \mathrm{Sym}(\mathcal{R}_\alpha)$$

with $\mathrm{Ker}(q) = \{c \mid (r)c = r \quad \forall r \in \mathcal{R}_\alpha\}$.

Proof. The map q is a natural homomorphism arising from the action of $C_V(\alpha)$ on $\mathcal{R}_\alpha$ proven in Corollary 4.2.5. Moreover, all elements of $\mathrm{Ker}(q)$ fix each of the repelling points of α as required. □

The Slope Map $\mathcal{S}$

We will now introduce the slope map $\mathcal{S}$. Later on, a direct analysis of $\mathcal{S}$ will help us determine the structure of the part of $C_V(\alpha)$ which is of immediate interest to us. The same slope map $\mathcal{S}$ was introduced in Bleak *et al.* in [5] but described using continuous analysis on a unit interval in which Cantor space can be embedded. However as we work with prefix substitutions instead, these are the terms with which we will represent the map $\mathcal{S}$.

To start with, we will need a couple of definitions to build on in order to define $\mathcal{S}$:

Definition 4.2.7. Let us define the length function $L : \{0,1\}^* \to \mathbb{N}_0$: for any $s \in \{0,1\}^*$, its length $L(s)$ is given by number of digits in the string s.

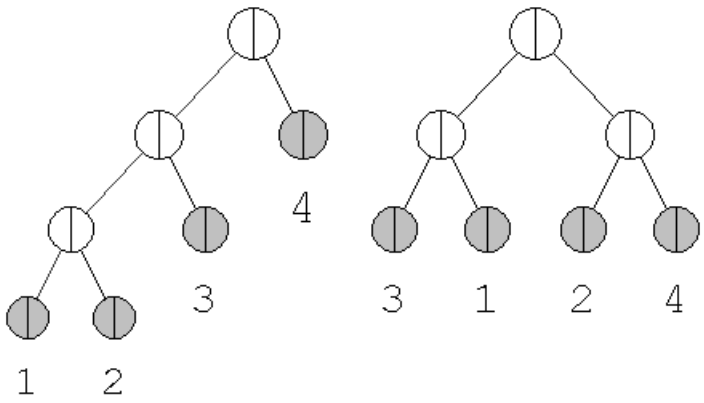
Definition 4.2.8. Suppose that $\alpha \in V$. Consider any $r \in \mathcal{R}_\alpha$. Then the function $f_r : C_V(\alpha) \to \mathbb{Z}$ is given by:

$$f_r(c) = L(\lambda) - L\big((\lambda)c\big)$$

where c is represented by the tree pair (D, R, t) and λ is the leaf of the domain tree D such that λ is a prefix of r.

We obviously have many choices for a tree pair representing an $\alpha \in V$, and so we will need to show that the map f_r is well-defined. Before we do that though, we will present an example of such a function to aid the visualisation of the proof of f_r being well–defined.

Example 4.2.9. Consider the following tree pair for $\alpha \in V$:



We identify the following leaf chains:

1. $000 \overset{\alpha}{\longmapsto} 01 \overset{\alpha}{\longmapsto} 00$ which is an R–chain;
2. $001 \overset{\alpha}{\longmapsto} 10$ which is an SS–chain;
3. $1 \overset{\alpha}{\longmapsto} 11$ which is an A–chain.

From the only R–chain we can then identify $\mathcal{R}_\alpha = \{00(0)^\infty, 01(0)^\infty\}$. Hence as $\alpha \in C_V(\alpha)$, we can compute the following:

$$\begin{aligned} f_{(0)^\infty}(\alpha) =& L(000) - L(01) = 3 - 2 = 1 \\ f_{01(0)^\infty}(\alpha) =& L(01) - L(00) = 2 - 2 = 0 \end{aligned}$$

Now we are prepared to prove:

Lemma 4.2.10. *Suppose that $\alpha \in V$. Consider any $r \in \mathcal{R}_\alpha$. Then the function f_r is well–defined.*

Proof. Suppose $c \in C_V(\alpha)$ and let c be represented by a tree pair (D, R, t). Recall by Corollary 3.2.31 that any tree pair representing α can be obtained by a finite number of simple augmentations and simple reductions from any other tree pair for α. We will show that if (D', R', t') is a simple augmentation from (D, R, t), then both tree pairs yield the same value $f_r(c)$. This will prove that $f_r(c)$ is independent of the choice of the tree pair.

Suppose that λ is the leaf of D which is a prefix for r. If the single augmentation was not performed on the leaf λ, then $\lambda \in L_{D'}$ and $f_r(c) = L(\lambda) - L\big((\lambda)c\big)$ in both cases. Otherwise, if the single augmentation occurred for λ, then only one of the leaves $\lambda 0$ and $\lambda 1$, of D', is a prefix for r. Say that λa for $a \in \{0, 1\}$ is the prefix for r. Then, using the tree pair (D', R', t'), the definition of $f_r(c)$ yields $f_r(c) = L(\lambda a) - L\big((\lambda a)c\big) = L(\lambda) + L(a) - L\big((\lambda)ca\big) = L(\lambda) + 1 - L\big((\lambda)c\big) - L(a) = L(\lambda) - L\big((\lambda)c\big)$. This is the same as when we used the tree pair (D, R, t), and hence the function f_r is well-defined. □

By now we have built all the definitions necessary for us to define the slope map $\mathcal{S}$:

Definition 4.2.11. Suppose that $\alpha \in V$. Define the function $\mathcal{S} : C_V(\alpha) \to \mathbb{Z}$ to be given by:

$$\mathcal{S}(c) = \sum_{r \in \mathcal{R}_\alpha} f_r(c)$$

We set $\mathcal{S}(C_V(\alpha)) = 0$ if $\mathcal{R}_\alpha = \{\}$.

We will want to show that $\mathcal{S}$, defined on $C_V(\alpha)$, is a group homomorphism. In order to do that, we will first prove the following lemma:

Lemma 4.2.12. *Suppose that $\alpha \in V$. For any $r \in \mathcal{R}_\alpha$ and any $c_1, c_2 \in C_V(\alpha)$ we have $f_r(c_1c_2) = f_r(c_1) + f_{(r)c_1}(c_2)$.*

Proof. Suppose that c_1 is represented by a tree pair (D_1, R_1, t_1), and λ_1 is a leaf of D_1 such that λ_1 is a prefix for r. Similarly, suppose that c_2 is represented by a tree pair (D_2, R_2, t_2), and λ_2 is a leaf of D_2 such that λ_2 is a prefix for $(r)c_1$. Note that $(r)c_1$ is another repelling point of α as the centraliser of α acts on the set of repelling points of α by Lemma 4.2.4. We are interested in where a prefix for r is being sent to under the action of c_1c_2. Recall that c_1 is a bijection from L_{D_1} to L_{R_1} and c_2 is a bijection from L_{D_2} to L_{R_2}. Recall also that by definition $f_r(c_1) = L(\lambda_1) - L\big((\lambda_1)c_1\big)$ and $f_{(r)c_1}(c_2) = L(\lambda_2) - L\big((\lambda_2)c_2\big)$. As $(\lambda_1)c_1$ and λ_2 are both prefixes for $(r)c_1$, then either $(\lambda_1)c_1 = \lambda_2$, or λ_2 is a proper prefix for $(\lambda_1)c_1$, or $(\lambda_1)c_1$ is a proper prefix for λ_2. Now we will discriminate between these three cases.

First suppose that $(\lambda_1)c_1 = \lambda_2$. In this case, $(\lambda_1)(c_1c_2) = ((\lambda_1)c_1)c_2 = (\lambda_2)c_2$. Therefore, $f_r(c_1c_2) = L(\lambda_1) - L\big((\lambda_1)(c_1c_2)\big) = L(\lambda_1) - L\big((\lambda_1)c_1\big) + L(\lambda_2) - L\big((\lambda_2)c_2\big) = f_r(c_1) + f_{(r)c_1}(c_2)$.

Otherwise, suppose that λ_2 is a proper prefix for $(\lambda_1)c_1$. This means that there is a positive integer k such that $(\lambda_1)c_1 = \lambda_2 a_1 a_2 \dots a_k$ for $a_1, \dots a_k \in \{0, 1\}$. Hence, there is a chain of augmentations from (D_2, R_2, t_2) to
(D_2', R_2', t_2'), which still represents c_2, such that $\lambda_2 a_1 \dots a_k \in L_{D_2'}$ and $\lambda_2 a_1 \dots a_k$ is a prefix for $(r)c_1$. In this case, $(\lambda_1)(c_1c_2) = ((\lambda_1)c_1)c_2 = (\lambda_2 a_1 \dots a_k)c_2$. Therefore, $f_r(c_1c_2) = L(\lambda_1) - L\big((\lambda_1)(c_1c_2)\big) = L(\lambda_1) - L\big((\lambda_2 a_1 \dots a_k)c_2\big) = L(\lambda_1) - L\big((\lambda_1)c_1\big) + L(\lambda_2 a_1 \dots a_k) - L\big((\lambda_2 a_1 \dots a_k)c_2\big)$
$= f_r(c_1) + f_{(r)c_1}(c_2)$.

Finally, suppose instead that $(\lambda_1)c_1$ is a proper prefix for λ_2. This means that there is a positive integer k such that $(\lambda_1)c_1 a_1 a_2 \dots a_k = \lambda_2$

for $a_1, \dots a_k \in \{0, 1\}$. Hence, there is a chain of augmentations from (D_1, R_1, t_1) to (D_1', R_1', t_1'), which still represents c_1, such that $\lambda_1 a_1 a_2 \dots a_k$ belongs to $L_{D_1'}$ and $\lambda_1 a_1 a_2 \dots a_k$ is a prefix for r. In this case,

$$(\lambda_1 a_1 a_2 \dots a_k)(c_1 c_2) = ((\lambda_1 a_1 a_2 \dots a_k)c_1)c_2 = \\ = ((\lambda_1)c_1 a_1 \dots a_k)c_2 = (\lambda_2)c_2.$$

Therefore,

$$f_r(c_1 c_2) = \\ = L(\lambda_1 a_1 \dots a_k) - L\big((\lambda_1 a_1 \dots a_k)(c_1 c_2)\big) = \\ = L(\lambda_1 a_1 \dots a_k) - L\big((\lambda_2)c_2\big) = \\ = L(\lambda_1 a_1 \dots a_k) - L\big((\lambda_1 a_1 \dots a_k)c_1\big) + L(\lambda_2) - L\big((\lambda_2)c_2\big) = \\ = f_r(c_1) + f_{(r)c_1}(c_2).$$

Thus, in all cases we conclude that $f_r(c_1 c_2) = f_r(c_1) + f_{(r)c_1}(c_2)$ as required. □

Now we can prove the following lemma:

Lemma 4.2.13. *Suppose that α is an infinite order element of V. Then $\mathcal{S} : C_V(\alpha) \to \mathbb{Z}$ is a group homomorphism.*

Proof. We need to prove that for any $c_1, c_2 \in C_V(\alpha)$, we have $\mathcal{S}(c_1 c_2) = S(c_1) + S(c_2)$. Recall from Lemma 4.2.12 that for any $r \in \mathcal{R}_\alpha$ and any $c_1, c_2 \in C_V(\alpha)$ we have $f_r(c_1 c_2) = f_r(c_1) + f_{(r)c_1}(c_2)$.

Thus, we can conclude that:

$$\mathcal{S}(c_1 c_2) = \sum_{r \in \mathcal{R}_\alpha} f_r(c_1 c_2) = \sum_{r \in \mathcal{R}_\alpha} (f_r(c_1) + f_{(r)c_1}(c_2)) = \\ \sum_{r \in \mathcal{R}_\alpha} f_r(c_1) + \sum_{r \in \mathcal{R}_\alpha} f_{(r)c_1}(c_2)$$

We recall that c_1 acts as a permutation on $\mathcal{R}_\alpha$ and hence:

$$\mathcal{S}(c_1 c_2) = \sum_{r \in \mathcal{R}_\alpha} f_r(c_1) + \sum_{r \in \mathcal{R}_\alpha} f_r(c_2) = \mathcal{S}(c_1) + \mathcal{S}(c_2)$$

This proves that $\mathcal{S}$ is a group homomorphism. □

Centraliser of a Connected Non–Periodic Flow Graph

Recall that our main results summarised in Theorem 4.1.4 and Theorem 4.1.5 refer to the centraliser of $\alpha \in V$, when α admits a single connected non–periodic flow graph component. In this subsection we show the result by Bleak *et al.* [5] that if α corresponds to a single connected non–periodic flow graph component then its centraliser decomposes as $C_V(\alpha) \cong \mathrm{Ker}(\mathcal{S}) \rtimes \mathbb{Z}$, with $|\mathrm{Ker}(\mathcal{S})| < \infty$. Our constructions in the following section will allow us to prove that each isomorphism type $K \rtimes \mathbb{Z}$ for a finite group K is realisable as a centraliser of some $\alpha \in V$.

To start with, we want to show that centraliser of any $\alpha \in V$ splits as a semidirect product of kernel and image of the slope map $\mathcal{S}$:

Lemma 4.2.14. *For a given $\alpha \in V$ such that α is of infinite order, there is a natural short exact sequence:*

$$0 \longrightarrow \mathrm{Ker}(\mathcal{S}) \longrightarrow C_V(\alpha) \longrightarrow \mathrm{Im}(\mathcal{S}) \longrightarrow 0$$

where $\mathrm{Im}(\mathcal{S}) \cong \mathbb{Z}$.

Proof. By Lemma 4.2.13 the function $\mathcal{S}$ is a group homomorphism from $C_V(\alpha)$ to $\mathbb{Z}$. Hence there is a natural short exact sequence as indicated and also $\mathrm{Im}(\mathcal{S})$ is a subgroup of $\mathbb{Z}$. Hence, if $\mathrm{Im}(\mathcal{S}) \neq \{0\}$ then $\mathrm{Im}(\mathcal{S}) \cong \mathbb{Z}$. We will now show that $\mathcal{S}(\alpha) \neq 0$. Suppose that (D, R, t) is a revealing pair for α and suppose that $((\lambda)\alpha^i)_{i=0}^k$ is the R–chain of a repeller λ, which exists by assumed properties of α. Let r be the repelling point underlying the repeller λ. Then we can compute

$$\sum_{r' \in \mathcal{O}_{\langle\alpha\rangle}(r)} f_{r'}(\alpha) = \sum_{i=0}^{k} f_{(r)\alpha^i}(\alpha) =$$

$$\sum_{i=0}^{k} (L((\lambda)\alpha^i) - L\big((\lambda)\alpha^{i+1}\big) =$$

$$L(\lambda) - L\big((\lambda)\alpha^k\big) > 0$$

We know that $L(\lambda) - L((\lambda)\alpha^k) > 0$ as λ is a repeller and $(\lambda)\alpha^k$ is its corresponding range of repulsion. The set $\mathcal{R}_\alpha$ is a union of disjoint orbits of repelling points under the action of α. $\mathcal{S}(\alpha)$ is the sum of the positive expressions as above over all orbits of repelling points. Hence, $\mathcal{S}(\alpha) > 0$ and thus $\mathrm{Im}(\mathcal{S}) \cong \mathbb{Z}$. □

Note that we do not necessarily have to have a $c \in C_V(\alpha)$ such that $\mathcal{S}(c) = 1$, i.e., $\mathcal{S}$ does not need to be onto.

We are trying to show that the group extension described in Lemma 4.2.14 is a split extension. In order to do that, we will first prove the following general lemma:

Lemma 4.2.15. *Let the groups K, G be such that the following is a short exact sequence:*

$$0 \longrightarrow K \longrightarrow G \xrightarrow{\psi} \mathbb{Z} \longrightarrow 0$$

Then $G \cong K \rtimes \mathbb{Z}$.

Proof. Let $z \in \psi^{-1}(1)$. Then $\mathbb{Z} \cong Z = \langle z \rangle \leq G$ and as $\mathrm{Ker}(\psi) = K$, $K \cap Z = \{1_G\}$. Now consider any $g \in G$. If $\psi(g) = k$ then g can be expressed as $g = (gz^{-k})z^k$. As $\psi(gz^{-k}) = \psi(g) + \psi(z^{-k}) = k - k = 0$, then $gz^{-k} \in K$ and so $G = KZ$. As $G = KZ$ and $K \cap Z = \{1_G\}$, we have $G \cong K \rtimes Z \cong K \rtimes \mathbb{Z}$ as required. □

Corollary 4.2.16. *Let $\alpha \in V$ such that $|\alpha| = \infty$. Then $C_V(\alpha) \cong Ker(\mathcal{S}) \rtimes \mathbb{Z}$.*

Proof. By Lemma 4.2.14 and Lemma 4.2.15. □

At this moment in order to proceed we need to introduce additional restrictions regarding the nature of the flow graph representing an element of V which we work with. We want to focus on the structure of a single non–periodic component, and for that we define:

Definition 4.2.17. Consider an $\alpha \in V$ and its flow graph. Suppose that the flow graph is connected and that almost all points from $\mathfrak{C}$ lie on infinite orbits under the action of $\langle \alpha \rangle$ (with the exception of important points $\mathcal{I}_\alpha$). Then the flow graph of α is called *connected non–periodic*, or *CNP flow graph*.

Note that the chains graph of α as in the definition above is connected and has no P–chains.

From now on we will require in our proofs that α has a CNP flow graph. We will now prove a technical proposition which will come in extremely handy in both this subsection and theoretical part of our construction in Section 4.3. The following proposition in more general form was first presented by Kassabov and Matucci in [19] where it was called Stair Algorithm. The proof as it stands is new:

Proposition 4.2.18 (Stair Algorithm)**.** *Let $\alpha \in V$ have CNP flow graph. Consider $r, s \in \mathcal{R}_\alpha$ and $c_1, c_2 \in C_V(\alpha)$. Suppose that:*

1. $(r)c_1 = s = (r)c_2$;
2. $f_r(c_1) = f_r(c_2)$.

Then $c_1 = c_2$.

Proof. First notice that $(r)c_1 = (r)c_2$ holds if and only if $(r)c_2c_1^{-1} = r$. Consider $f_r(c_2c_1^{-1})$. By Lemma 4.2.12 we have $f_r(c_2c_1^{-1}) = f_r(c_2) + f_s(c_1^{-1})$. As $(r)c_1 = s$ we have $0 = f_r(e) = f_r(c_1c_1^{-1}) = f_r(c_1) + f_s(c_1^{-1})$ which implies that $f_s(c_1^{-1}) = -f_r(c_1)$. Therefore $f_r(c_2c_1^{-1}) = f_r(c_2) + f_s(c_1^{-1}) = f_r(c_2) - f_r(c_1) = 0$, by the initial assumption. Let $c = c_2c_1^{-1}$. We will prove that if $(r)c = r$ and $f_r(c) = 0$ for $c \in C_V(\alpha)$, then $c = e$. This will prove that $c_1 = c_2$.

Let (D_c, R_c, t_c) be a tree pair for c. Let λ be the leaf of D_c which overlies r. Then as $f_r(c) = L(\lambda) - L((\lambda)c) = 0$, we know that $(\lambda)c$ is a word with the same length as λ. But note that $(r)c = r$ and so both λ and $(\lambda)c$ are prefixes for r. As they have the same length, we must have $\lambda = (\lambda)c$.

By Theorem 3.2.70 and Lemma 3.4.9 there is a revealing tree pair $(D_\alpha, R_\alpha, t_\alpha)$ representing α such that the repelling point r underlies a repeller λ_0, say. Let $(\lambda_i)_{i=0}^k$ be the R–chain of this tree pair corresponding to the repelling point r. We will first show that c fixes the range of repulsion λ_k and then that it fixes all leaves of this R–chain. Consider $(\lambda_k)\alpha^{-mk}$ for the smallest possible non-negative integer m, such that λ is a prefix of $(\lambda_k)\alpha^{-mk}$. Note that such m exists as $\lambda_0 = (\lambda_k)\alpha^{-k} = \lambda_k\Gamma$ for the spine Γ of $r = \lambda_k(\Gamma)^\infty$, and so $(\lambda_k)\alpha^{-mk} = \lambda_k(\Gamma)^m$. As λ is a prefix of $(\lambda_k)\alpha^{-mk}$ and λ is fixed by c, the word $(\lambda_k)\alpha^{-mk}$ is also fixed by c, namely $((\lambda_k)\alpha^{-mk})c = (\lambda_k)\alpha^{-mk}$. As $c \in C_V(\alpha)$, c commutes with all powers of α. Therefore:

$$(\lambda_k)\alpha^{-mk} = [(\lambda_k)\alpha^{-mk}]c = [(\lambda_k)c]\alpha^{-mk} \implies \lambda_k = (\lambda_k)c.$$

This means that c fixes the range of repulsion of the initial R–chain $(\lambda_i)_{i=0}^k$. Now consider any non-negative integer l such that $l \leq k$. Then:

$$\lambda_l = (\lambda_k)\alpha^{l-k} = [(\lambda_k)c]\alpha^{l-k} = [(\lambda_k)\alpha^{l-k}]c = (\lambda_l)c.$$

Therefore c fixes all leaves from the R–chain $(\lambda_i)_{i=0}^{k}$, and so all the points from Cantor space underlying these leaves are fixed by c.

Now we will show that c fixes all leaves from any SS–chains of the tree pair $(D_\alpha, R_\alpha, t_\alpha)$ which start with a source underlying the range of repulsion leaf λ_k. Consider any such chain $(\lambda'_i)_{i=0}^{k'}$. Then as by assumption λ'_0 is a descendant of λ_k, we have $(\lambda'_0)c = \lambda'_0$ as well. Moreover, for any non-negative integer l such that $l \leq k'$ we have:

$$\lambda'_l = (\lambda'_0)\alpha^l = [(\lambda'_0)c]\alpha^l = [(\lambda'_0)\alpha^l]c = (\lambda'_l)c.$$

Therefore c fixes all leaves from the SS–chain $(\lambda'_i)_{i=0}^{k'}$, and so all the points from Cantor space underlying these leaves are fixed by c.

Next we will prove that c fixes the domain of attraction leaf of each A–chain of the tree pair $(D_\alpha, R_\alpha, t_\alpha)$ which overlies a sink which is fixed by c. From this we will prove that c fixes all leaves of this A–chain. Consider such an A–chain $(\lambda''_i)_{i=0}^{k''}$. Say that $\lambda'_{k'}$ is a sink such that $(\lambda'_{k'})c = \lambda'_{k'}$ and λ''_0 is a prefix of $\lambda'_{k'}$. Also let a be the attracting point of α underlying the attractor $\lambda''_{k''}$. Recall the tree pair (D_c, R_c, t_c) for c. Let λ' be the leaf of the tree D_c overlying a. We will now show that we must have $(\lambda')c = \lambda'$. As previously, there must be a non-negative integer m' such that $(\lambda''_0)\alpha^{m'k''}$ is a descendant of λ'. Then, $(\lambda'_{k'})\alpha^{m'k''}$ also has to be a descendant of λ'. Recall that $\lambda'_{k'}$ is fixed by c. Then:

$$(\lambda'_{k'})\alpha^{m'k''} = [(\lambda'_{k'})c]\alpha^{m'k''} = [(\lambda'_{k'})\alpha^{m'k''}]c.$$

Thus $(\lambda'_{k'})\alpha^{m'k''}$ is also fixed by c. Now consider that $(\lambda')c = \lambda''$ for some word λ''. Let $(\lambda'_{k'})\alpha^{m'k''} = \lambda' w$ for some word w. Then we have to have $(\lambda' w)c = \lambda'' w$. But as $\lambda' w$ is fixed by c, $\lambda' = \lambda''$ and hence c fixes λ'. Therefore, it also fixes $(\lambda''_0)\alpha^{m'k''}$. As previously we have:

$$(\lambda''_0)\alpha^{m'k''} = [(\lambda''_0)\alpha^{m'k''}]c = [(\lambda''_0)c]\alpha^{m'k''}.$$

Hence, we have $(\lambda''_0)c = \lambda''_0$. Therefore c fixes the domain of attraction λ''_0 as required. Moreover, for all non-negative integers $l \leq k''$ we have:

$$\lambda''_l = (\lambda''_0)\alpha^l = [(\lambda''_0)c]\alpha^l = [(\lambda''_0)\alpha^l]c = (\lambda''_l)c.$$

Thus c fixes all leaves in the A–chain $(\lambda''_i)_{i=0}^{k''}$.

Therefore whenever we have a sink under a domain of attraction such

that the sink is fixed by c, this domain of attraction and all its images under positive powers of α are also fixed by c.

Similarly we can prove that for any sink underlying a domain of attraction which is fixed by c, the whole SS–chain finishing at that sink is fixed by c. Also similarly for any range of repulsion overlying a source fixed by c, this range of repulsion and all its images under negative powers of α are fixed by c.

Recall that the flow graph of α is connected (and non–periodic). Hence each R–chain, SS–chain and A–chain is connected to our initial R–chain via a sequence of flow lines. Hence, all their leaves are fixed by c. As any point x of Cantor space underlies a leaf of some of these leaf chains, $(x)c = x$ for all $x \in \mathfrak{C}$. Hence:

$$e = c_2 c_1^{-1} \implies c_2 = c_1$$

This proves the lemma. □

We continue with analysis of the structure of the centraliser of an α which has the CNP flow graph. From now on, we are building a proof that the group $\mathrm{Ker}(\mathcal{S})$ is finite.

Lemma 4.2.19. *Let $\alpha \in V$ have the CNP flow graph and recall the natural homomorphism $q : C_V(\alpha) \longrightarrow \mathrm{Sym}(\mathcal{R}_\alpha)$ introduced in 4.2.6 and corresponding to the action of $C_V(\alpha)$ on $\mathcal{R}_\alpha$. Then, $\mathrm{Ker}(q) \cong \mathbb{Z}$. In particular, there is a short exact sequence:*

$$0 \longrightarrow \mathbb{Z} \longrightarrow C_V(\alpha) \xrightarrow{q} Q \longrightarrow 0$$

such that Q is a finite group.

Proof. We have that $\mathrm{Im}(q) = Q \leq \mathrm{Sym}(\mathcal{R}_\alpha)$ and so Q is finite. If we now prove that $\mathrm{Ker}(q) \cong \mathbb{Z}$, then there is a short exact sequence as required.

For simplicity of notation, let $M = \mathrm{Ker}(q)$. We will consider the restricted map $f_r|_{\mathrm{M}} : M \longrightarrow \mathbb{Z}$. We will show that for any given $r \in \mathcal{R}_\alpha$ the function $f_r|_{\mathrm{M}}$ is an injective homomorphism. This will imply that M is isomorphic to a subgroup of $\mathbb{Z}$. Finally we will show that $|M| > 1$ which will prove that M is non-trivial, and hence isomorphic to $\mathbb{Z}$.

First of all, recall by Lemma 4.2.12 that for each $r \in \mathcal{R}_\alpha$ and any $c_1, c_2 \in C_V(\alpha)$ we have $f_r(c_1 c_2) = f_r(c_1) + f_{(r)c_1}(c_2)$. Thus, if $c_1, c_2 \in M$,

then in particular $(r)c_1 = r$ and so $f_r(c_1c_2) = f_r(c_1) + f_r(c_2)$. Hence the restricted map $f_r|_{\mathrm{M}}$ is a group homomorphism.

Secondly, suppose that $c_1, c_2 \in M$ are such that $f_r(c_1) = f_r(c_2)$. As $c_1, c_2 \in M$, it means that both c_1 and c_2 fix the set $\mathcal{R}_\alpha$ pointwise. Hence, as α has the CNP flow graph, by Proposition 4.2.18 we need to have $c_1 = c_2$. This implies that the homomorphism $f_r|_{\mathrm{M}}$ is injective.

Finally, let $k = |Q|$ and $q(\alpha) = \sigma \in Q$. Then $\alpha^k \in M$ as $q(\alpha^k) = \big(q(\alpha)\big)^k = \sigma^k = 1_Q$, because the order of σ must divide the order of Q. Moreover, $|\alpha| = \infty$ as α has important points. Hence $1_V, \alpha^k \in M$, which implies that $|M| > 1$.

This proves that $\mathrm{Ker}(q) = M \cong \mathbb{Z}$ as required. □

We will now prove a general result about structure of a group which admits certain extensions, and apply it immediately afterwards to our case of interest.

Lemma 4.2.20. *Let G, Q, K, M be groups where $M \cong \mathbb{Z}$ and $|Q| = k < \infty$. Assume that there are two short exact sequences as follows:*

$$0 \longrightarrow M \xrightarrow{i_M} G \xrightarrow{q} Q \longrightarrow 0$$

and

$$0 \longrightarrow K \xrightarrow{i_K} G \xrightarrow{\psi} \mathbb{Z} \longrightarrow 0$$

Then $|K| < \infty$.

Proof. Without loss of generality we may assume that the maps i_K and i_M are the inclusion maps and so we can form the intersection $K \cap M$ in G. We aim to show that K is a torsion subgroup. Once we prove that, as $M \cong \mathbb{Z}$, we can deduce that $K \cap M = \{1_G\}$. As $M = \mathrm{Ker}(q)$, this implies that the restricted map $q|_K : K \longrightarrow Q$ is an injection. Thus as $|Q| < \infty$, $|K| = |q(K)| \leq |Q| < \infty$.

Let $z \in \psi^{-1}(1)$ and $Z = \langle z \rangle \leq G$. Recall that also $M \leq G$. As $Z \cong \mathbb{Z}$, in particular Z is an abelian group and so all its subgroups are normal in it. Hence we may consider the quotient $Z/Z \cap M$. By the second isomorphism theorem, $Z/Z \cap M \cong ZM/M$. Then $Z/Z \cap M \cong ZM/M \leq G/M \cong Q$. As $|Q| < \infty$, $|Z : Z \cap M| < \infty$. In particular, $|Z \cap M| > 1$ and so as $M \cong \mathbb{Z}$, $|M : M \cap Z| < \infty$. Recall that $|Q| = k$. Hence we know that for all $g \in G$ we have $g^k \in M$. Let $m = |M : M \cap Z|$. Then we also conclude that $g^{km} \in M \cap Z$. Now if in addition $g \in K$ then

$g^{km} \in M \cap Z \cap K \leq K \cap Z = \{1_G\}$, as $G \cong K \rtimes Z$ by Lemma 4.2.15. Therefore, K is a torsion subgroup of finite exponent.

Thus by the discussion at the beginning of the proof we conclude that $|K| < \infty$ as required. □

Corollary 4.2.21. *Let $\alpha \in V$ have a CNP flow graph. Recall the function $\mathcal{S}$ from Definition 4.2.11. Then $|Ker(\mathcal{S})| < \infty$.*

Proof. Recall Lemma 4.2.14 and Lemma 4.2.19. Apply them to Lemma 4.2.20 for $G = C_V(\alpha)$ and $\psi = \mathcal{S}$. □

We will also present our new alternative proof for Corollary 4.2.21 which does not use Lemma 4.2.20. We felt that both proofs should be included for at least two reasons. First of all, Lemma 4.2.20 is more general and it was only after careful examination of it that we understood a construction that would work as an alternative proof. On the other hand, precisely because Lemma 4.2.20 is more general, it does not highlight the particular properties of the exact sequences with which we work. Therefore, we present the following:

Lemma 4.2.22. *Consider an $\alpha \in V$ with a CNP flow graph. Recall the short exact sequences from Lemma 4.2.14 and Lemma 4.2.19:*

$$0 \longrightarrow M \xrightarrow{i_M} C_V(\alpha) \xrightarrow{q} Q \longrightarrow 0$$

and

$$0 \longrightarrow K \xrightarrow{i_K} C_V(\alpha) \xrightarrow{\mathcal{S}} \mathbb{Z} \longrightarrow 0$$

Recall that Q is finite and $M \cong \mathbb{Z}$. Then K is a finite group.

Proof. We will show that $K \cap M = \{1_{C_V(\alpha)}\}$. This will imply that the restricted map $q|_K$ is an isomorphism, and so $K \cong q(K) \leq Q$, which implies $|K| \leq |Q| < \infty$.

Recall from the proof of Lemma 4.2.19 that $\alpha^{|Q|} \in M$. As $M \cong \mathbb{Z}$, say that $M = \langle \beta \rangle$. Then there is a non–zero integer s such that $\beta^s = \alpha^{|Q|}$. Now recall from the proof of Lemma 4.2.14 that $\mathcal{S}(\alpha) > 0$. Hence we have:

$$0 < |Q| \cdot \mathcal{S}(\alpha) = \mathcal{S}(\alpha^{|Q|}) = \mathcal{S}(\beta^s) = s \cdot \mathcal{S}(\beta)$$

Therefore, for any integer l, $\mathcal{S}(\beta^l) = 0$ if and only if $l = 0$. This implies that $K \cap M = \{1_{C_V(\alpha)}\}$ as required and hence by the discussion at the beginning of the proof, K is finite. □

Finally, for a better understanding of elements of $\mathrm{Ker}(\mathcal{S})$ in our case of interest, we present the following lemma:

Lemma 4.2.23. *Let $\alpha \in V$ and let α have the CNP flow graph. Then $Ker(\mathcal{S})$ is precisely the set of torsion elements of $C_V(\alpha)$.*

Proof. Let T be the set of torsion elements of $C_V(\alpha)$ and let $K = \mathrm{Ker}(\mathcal{S})$. By Lemma 4.2.22, K is finite and so $K \subseteq T$. Conversely, let $t \in T$. This implies that there is a positive integer s such that $t^s = 1_{C_V(\alpha)}$. As $\mathcal{S}$ is a group homomorphism, we have $0 = \mathcal{S}(t^s) = s \cdot \mathcal{S}(t)$ which implies that $\mathcal{S}(t) = 0$. This is equivalent to saying that $t \in K$ and thus $T \subseteq K$. Therefore, $T = K$ as required. □

4.3 Elements with a Prescribed Centraliser

The new work presented in this section combines new theory derivations with some hands–on constructions. We start by presenting the details of the action of the torsion part of the centraliser of an element $\alpha \in V$ such that α admits the CNP flow graph, on its repelling points. The implications of these details in the first place highlight conditions which a successful construction of elements with prescribed centralisers need to possess. We subsequently present two constructions which target questions (2) and (3) from [5] respectively. The answers to these questions are stated in Theorems 4.1.4 and 4.1.5. Full implications of these together with the work from [5] are summarised in Theorem 4.1.2. Then, the previously established theory allows us to conclude that there are no more elements in the centraliser's torsion subgroup or entire centraliser respectively. The core idea of the constructions is to embed a graph with a high degree of symmetry into the flow graphs of elements which we are designing. We chose to use Cayley graphs for that purpose. Finally we present examples of an application of each of the constructions. Note that it is only verifiable after presenting all of the theory that the centralisers are not bigger than stated in the examples.

Action of K on $\mathcal{R}_\alpha$

Let $\alpha \in V$ have a CNP flow graph and K be the torsion subgroup of its centraliser. We will now perform a detailed analysis of the action of K on the set of repelling points of α. The properties shown in this subsection

are crucial to proving that the torsion subgroup of the centraliser of the element designed in Constructions 4.3.7 and 4.3.13 is not bigger than we claim.

One of the properties of K which we can deduce from the previous section is that K acts faithfully on the repelling points of α.

Corollary 4.3.1. *Consider $\alpha \in V$ with a CNP flow graph. Then, $Ker(\mathcal{S})$ acts faithfully on the set $\mathcal{R}_\alpha$.*

Proof. By the proof of Lemma 4.2.22, $K \cong q(K) \leq Q \leq \mathrm{Sym}(\mathcal{R}_\alpha)$. □

However, with a bit more work, we can prove an even stronger property about the action of K on $\mathcal{R}_\alpha$, namely that every non-trivial element of K moves each of the repelling points of α. Below we will prove this fact and discuss its implications. We will also point out how it helped us come up with an idea for our constructions.

Lemma 4.3.2. *Let $\alpha \in V$ have a CNP flow graph. Let K be the torsion of the centraliser of α. Then if $(r)\beta = r$ for some $r \in \mathcal{R}_\alpha$ and some $\beta \in K$, then $\beta = 1_K$.*

Proof. Suppose that $(r)\beta = r$ for some $r \in \mathcal{R}_\alpha$ and some $\beta \in K$. First of all, we will show that the restricted map $f_r|_{\langle\beta\rangle}$ is a group homomorphism. Recall from Lemma 4.2.12 that for any $c_1, c_2 \in C_V(\alpha)$ we have $f_r(c_1c_2) = f_r(c_1) + f_{(r)c_1}(c_2)$. Hence as $(r)\beta = r$ for all $i, j \in \mathbb{Z}$ we have:

$$f_r(\beta^{i+j}) = f_r(\beta^i) + f_{(r)\beta^i}(\beta^j) = f_r(\beta^i) + f_r(\beta^j).$$

Hence, $f_r|_{\langle\beta\rangle} : \langle\beta\rangle \longrightarrow \mathbb{Z}$ is a group homomorphism. Now as β is an element of torsion K, we have that $\beta^{|K|} = 1_K$. Hence, $0 = f_r(1_K) = f_r(\beta^{|K|}) = |K| \cdot f_r(\beta)$, which implies that $f_r(\beta) = 0$. But by Proposition 4.2.18, as $(r)\beta = r = (r)1_K$ and $f_r(\beta) = f_r(1_K)$, we must have $\beta = 1_K$. □

Corollary 4.3.3. *Let $\alpha \in V$ have the CNP flow graph. Let K be the torsion of the centraliser of α and $r \in \mathcal{R}_\alpha$. Then $\mathrm{Stab}_K(r) = \{1_K\}$, $|K| = |\mathcal{O}_K(r)|$, $|K|$ divides $|\mathcal{R}_\alpha|$ and the action of K on $\mathcal{O}_K(r)$ is regular.*

Proof. By Lemma 4.3.2, if $\beta \in K$ stabilises r, then $\beta = 1_K$ and so $\mathrm{Stab}_K(r) = \{1_K\}$ for all $r \in \mathcal{R}_\alpha$. By the orbit-stabiliser theorem $|K| = |\mathcal{O}_K(r)| \cdot |\mathrm{Stab}_K(r)|$ but as $|\mathrm{Stab}_K(r)| = 1$ we have $|K| = |\mathcal{O}_K(r)|$. As

orbits of repelling points under the action of K are of size $|K|$ each and as they partition $\mathcal{R}_\alpha$ into disjoint sets, $|K|$ divides $|\mathcal{R}_\alpha|$. Finally, K is transitive on $\mathcal{O}_K(r)$ by definition. We will now show that for any $r, r' \in \mathcal{O}_K(r)$ there is precisely one $\beta \in K$ such that $(r)\beta = r'$. This will prove that the action of K on $\mathcal{O}_K(r)$ is regular. Suppose then that there are $\beta_1, \beta_2 \in K$ such that $(r)\beta_1 = (r)\beta_2$. This implies that $(r)\beta_1\beta_2^{-1} = r$ and by Lemma 4.3.2 it proves that $\beta_1 = \beta_2$. □

Observation 4.3.4. Suppose that $\alpha \in V$ has a CNP flow graph and that K is the torsion subgroup of centraliser of α. Equivalent results linking K and $\mathcal{R}_\alpha$ hold as well for $\mathcal{A}_\alpha$. In particular, for all $r \in \mathcal{R}_\alpha$ and for all $a \in \mathcal{A}_\alpha$, $|\mathcal{O}_K(r)| = |K| = |\mathcal{O}_K(a)|$. Thus the action of K partitions the set of attracting points of α into orbits of the same size as the orbits of the repelling points. Note that this does not imply that the number of the repelling points is equal to the number of attracting points. We also conclude that if $\beta \in K$ stabilises any $p \in \mathcal{I}_\alpha$, then β is the identity map.

The remark below explains how we use the information above for the new constructions, and how we know that we have identified the full torsion subgroup of the centraliser.

Remark 4.3.5. Suppose that we can construct an α belonging to V with CNP flow graph such that the torsion subgroup K of centraliser has a subgroup K' which is isomorphic to a chosen finite group A. Then we have the following inequalities:

$$|A| \leq |K| = |\mathcal{O}_K(r)| \leq |\mathcal{R}_\alpha|$$

for any $r \in \mathcal{R}_\alpha$. Hence, if we can construct α such that $|A| = |\mathcal{R}_\alpha|$, we can conclude that $A \cong K' = K$.

Construction of α with Prescribed Torsion Subgroup of its Centraliser

For a given finite group A, we can now proceed to construct an $\alpha \in V$ with a CNP flow graph and with torsion subgroup K of the centraliser such that $K \cong A$. This is where the connection with the Cayley graph of A and flow graph of α will become apparent. Together with Lemma 4.3.10, the construction will imply Theorem 4.1.4.

Note that we are going to use standard notation A for a generic finite group in our constructions. The reason to use A in Theorems 4.1.2, 4.1.4

and 4.1.5 was to remain consistent with the notation from Bleak *et al.* in [5].

Remark 4.3.6. We will construct a general example of an α with all its attracting and repelling points being stationary, and with one orbit of repelling points and one orbit of attracting points under the action of K.

Construction 4.3.7. Let A be any finite group. We now construct a revealing pair (D, R, t) for α:

1. Let $h_1 = g_1 = 1_A$. Enumerate the elements of $A = \{g_1, \ldots, g_{|A|}\}$. Pick a generating set $\{h_2, \ldots, h_t\}$ for A for some integer $t \geq 1$ (assume that the generating set is empty if A is trivial), and denote $S = \{h_1, \ldots, h_t\}$.

2. Pick a finite subtree of the infinite rooted binary tree $\mathcal{T}$ with precisely $2|A|$ leaves, and call it T_0.

3. Pick two finite subtrees of the infinite rooted binary tree $\mathcal{T}$ with precisely $1 + |S|$ leaves each, and call them T_r and T_a respectively.

4. Label $|S|$ of the leaves of T_r by the elements of S, and similarly, for T_a. The leaves which remain unlabelled will correspond to a repeller and an attractor in the case of T_r and T_a, respectively. For clarity of notation, label them both with a symbol h_0. For each integer i such that $0 \leq i \leq t$ let the leaf of T_r labelled with h_i have address Γ_{h_i} in this tree. Similarly, let the leaf of T_a labelled with h_i have address Γ'_{h_i} in this tree.

5. Pick $|A|$ leaves of T_0 and label them with the elements of A. Say that for each $g \in A$ the address of the leaf of T_0 labelled with g is λ_g. To create the tree D we attach $|A|$ copies of the labelled tree T_r to the tree T_0 by identifying the root of a copy of the tree T_r to each of these labelled leaves of T_0. Denote the copy of T_r which is now rooted at the leaf of T_0 with address λ_g as $T_r(g)$.

6. Consider the unlabelled tree T_0 again. Label the common leaves of the trees D and T_0 (namely those leaves of T_0 which were not picked in the process above) with elements of A. Say that for each $g \in A$ the address of the leaf of T_0 labelled with g is λ'_g. To create the tree R we attach $|A|$ copies of the tree T_a to the tree T_0 by identifying a root of a copy of the tree T_a to each of these labelled

leaves of T_0. Denote the copy of T_a which is now rooted at the leaf of T_0 with address λ'_g as $T_a(g)$.

7. Now establish the flow lines. For each $g \in A$ and $j \in \{1, \ldots, t\}$, the leaf $\lambda_g \Gamma_{h_j}$ of the tree D will be mapped to the leaf $\lambda'_{gh_j}\Gamma'_{h_j}$ of the tree R. Also, as mentioned earlier, the leaf $\lambda_g\Gamma_{h_0}$ of the tree D (repeller) is mapped to the leaf λ_g of the tree R. Similarly, the leaf λ'_g of the tree D, is mapped to the leaf $\lambda'_g\Gamma'_{h_0}$ of the tree R (attractor).

Lemma 4.3.8. *The flow graph of α from Construction 4.3.7 is a CNP graph.*

Proof. We will consider the flow graph of α. We will show that it is CNP. First of all α has a non-empty set $\mathcal{R}_\alpha = \{\lambda_g(\Gamma_{h_0})^\infty \mid g \in A\}$ of repelling points and non-empty set $\mathcal{A}_\alpha = \{\lambda'_g(\Gamma'_{h_0})^\infty \mid g \in A\}$ of attracting points. Hence if we show that the flow graph is connected, we can conclude that it is CNP.

The flow graph has flow lines going out of its vertices for every descendant of every address of the form λ_g being mapped somewhere by α and has flow lines coming into its vertices for every descendant of every address of the form λ'_g being an image of the map α.

We will show that for all $g_i, g_j \in A$ the vertex of the flow graph representing the repelling point $\lambda_{g_i}(\Gamma_{h_0})^\infty$ is connected to the vertex of the flow graph representing the attracting point $\lambda'_{g_j}(\Gamma'_{h_0})^\infty$.

This is because $g_i^{-1}g_j = h_{i_1}h_{i_2}\ldots h_{i_r}$ for some $h_{i_1}, \ldots, h_{i_r} \in S\backslash\{h_1\}$ for some non-negative integer r, and so:

$$\underbrace{\lambda_{g_i}\Gamma_{h_{i_1}}}_{\lambda_{g_i}} \xrightarrow{\alpha} \underbrace{\lambda'_{g_ih_{i_1}}\Gamma'_{h_{i_1}} \quad \lambda'_{g_ih_{i_1}}\Gamma'_{h_1}}_{\lambda'_{g_ih_{i_1}}} \xleftarrow{\alpha} \underbrace{\lambda_{g_ih_{i_1}}\Gamma_{h_1} \quad \lambda_{g_ih_{i_1}}\Gamma_{h_{i_2}}}_{\lambda_{g_ih_{i_1}}} \xrightarrow{\alpha} \ldots$$

$$\ldots \xrightarrow{\alpha} \underbrace{\lambda'_{g_ih_{i_1}\ldots h_{i_r}}\Gamma'_{h_{i_r}} = \lambda'_{g_j}\Gamma'_{h_{i_r}}}_{\lambda'_{g_j}}$$

Moreover, if $g_i = g_j$ (and so i=j), we have $g_i^{-1}g_j = h_1$ and

$$\underbrace{\lambda_{g_i}\Gamma_{h_1}}_{\lambda_{g_i}} \xrightarrow{\alpha} \underbrace{\lambda'_{g_ih_1}\Gamma'_{h_1} = \lambda'_{g_i}\Gamma'_{h_1}}_{\lambda'_{g_j}}$$

as required.

Thus α has a CNP flow graph. □

We will now demonstrate how we want to embed A into V so that the image of the embedding is the full torsion subgroup K of the centraliser of α in V.

Lemma 4.3.9. *Consider T_0 from Construction 4.3.7 and the corresponding finite group A. We have a faithful action of A on the leaves of T_0:*

$$\phi : A \hookrightarrow \mathrm{Sym}(\{\lambda_g, \lambda'_g \mid g \in A\}) \leq V.$$

which is defined by:

$$(\lambda_{g'})\phi(g) = \lambda_{g^{-1}g'}$$
$$(\lambda'_{g'})\phi(g) = \lambda'_{g^{-1}g'}$$

for all $g, g' \in A$.

Proof. We will show that $\phi(A)$ acts on the set L_{T_0} which will imply that ϕ is a homomorphism. For all $g', g_1, g_2 \in A$:

$$\begin{aligned}
(\lambda_{g'})\phi(e) &= \lambda_{e^{-1}g'} = \lambda_{g'} \\
(\lambda'_{g'})\phi(e) &= \lambda'_{e^{-1}g'} = \lambda'_{g'} \\
(\lambda_{g'})\phi(g_1g_2) &= \lambda_{(g_1g_2)^{-1}g'} = \lambda_{g_2^{-1}g_1^{-1}g'} = \\
&= (\lambda_{g_1^{-1}g'})\phi(g_2) = \big((\lambda_{g'})\phi(g_1)\big)\phi(g_2) \\
(\lambda'_{g'})\phi(g_1g_2) &= \lambda'_{(g_1g_2)^{-1}g'} = \lambda'_{g_2^{-1}g_1^{-1}g'} = \\
&= (\lambda'_{g_1^{-1}g'})\phi(g_2) = \big((\lambda'_{g'})\phi(g_1)\big)\phi(g_2)
\end{aligned}$$

Thus we conclude that ϕ is a homomorphism. Additionally, the action is faithful as in this case left multiplication by an inverse element has trivial effect if and only if we are multiplying by an identity. Hence ϕ is an embedding. Therefore, we will now regard $\phi(A)$ as a subgroup of V. □

Lemma 4.3.10. *Let α be defined by a tree pair (D, R, t) created via Construction 4.3.7. Then for each $g \in A$, the action of $\phi(g)$ defined in Lemma 4.3.9 commutes with the action of α.*

Proof. Pick any leaf of D. For all $g, g' \in A$ and all $h_j \in S$ we have three cases to check:

1.
$$\big((\lambda_{g'}\Gamma_{h_0})\alpha\big)\phi(g) = (\lambda_{g'})\phi(g) = \lambda_{g^{-1}g'} =$$
$$= (\lambda_{g^{-1}g'}\Gamma_{h_0})\alpha = \big((\lambda_{g'}\Gamma_{h_0})\phi(g)\big)\alpha$$

2.
$$\big((\lambda_{g'}\Gamma_{h_j})\alpha\big)\phi(g) = (\lambda'_{g'h_j}\Gamma'_{h_j})\phi(g) = (\lambda'_{g^{-1}g'h_j}\Gamma'_{h_j}) =$$
$$= (\lambda_{g^{-1}g'}\Gamma_{h_j})\alpha = \big((\lambda_{g'}\Gamma_{h_j})\phi(g)\big)\alpha$$

3.
$$\big((\lambda'_{g'})\alpha\big)\phi(g) = (\lambda'_{g'}\Gamma'_{h_0})\phi(g) = \lambda'_{g^{-1}g'}\Gamma'_{h_0}$$
$$= (\lambda'_{g^{-1}g'})\alpha = \big((\lambda'_{g'})\phi(g)\big)\alpha$$

Hence, each element of $\phi(A)$ centralises α. □

Corollary 4.3.11. *Consider α constructed from a finite group A in Construction 4.3.7. The torsion subgroup K of $C_V(\alpha)$ is isomorphic to A. This proves our Theorem 4.1.4.*

Proof. By Lemma 4.3.8 the flow graph of the constructed α is CNP. By Lemma 4.3.9 ϕ is an embedding of A into V. In Lemma 4.3.10 we showed that all elements of $\phi(A)$ commute with α. Hence, as all elements of A are of finite order, $\phi(A)$ is a subgroup of K, which is the set of all torsion elements of the centraliser of α. Then as α has precisely $|A|$ repelling points, by Remark 4.3.5, we have $A \cong \phi(A) = K$. □

Remark 4.3.12. We will now highlight the connection of the flow graph of α to the right Cayley graph of A with generating set $\{h_2, \ldots, h_t\}$.

Recall that the flow graph of α is CNP. It has the set of vertices being orbits of important points of α under the action of α. As for all $g \in A$ we have $(\lambda_g\Gamma_{h_0})\alpha = \lambda_g$, all the repelling points of α are stationary. Similarly, as for all $g \in A$ we have $(\lambda'_g)\alpha = \lambda'_g\Gamma'_{h_0}$, all the attracting points of α are stationary. Hence the set of vertices of the flow graph for α is

$\mathcal{I}_\alpha$. Then, for all $g, g' \in A$ and for all $h \in S$, there is a unique flow line, namely an edge of the flow graph, from the repelling point $\lambda_{g'}(\Gamma_{h_0})^\infty$ to the attracting point $\lambda'_{g'h}(\Gamma'_{h_0})^\infty$, with the label $\lambda_g\Gamma_h - \lambda'_{gh}\Gamma'_h$. Replace this label with label h. Now, consider the effect of $\phi(g)$ on this flow graph. The repelling point $\lambda_{g'}(\Gamma_{h_0})^\infty$ is sent to the repelling point $\lambda_{g^{-1}g'}(\Gamma_{h_0})^\infty$ and the attracting point $\lambda'_{g'h}(\Gamma'_{h_0})^\infty$ is sent to the attracting point $\lambda'_{g^{-1}g'h}(\Gamma'_{h_0})^\infty$. Notice that there is still an edge with label h from our image $\lambda_{g^{-1}g'}(\Gamma_{h_0})^\infty$ to our image $\lambda'_{g^{-1}g'h}(\Gamma'_{h_0})^\infty$.

Finally, for each $g \in A$, collapse the edge with label $h_1 = e$ between vertices of the flow graph denoted by $\lambda_g(\Gamma_{h_0})^\infty$ and $\lambda'_g(\Gamma'_{h_0})^\infty$, and call the new vertex g. We then recognise a right Cayley graph of A on generators $S\backslash\{h_1\}$, with the set of vertices A and edges labelled by elements of $S\backslash\{h_1\}$.

Construction 4.3.7 was inspired precisely by high degree of symmetry of Cayley graphs. The group A represents automorphisms group of its right Cayley graph. In its realisation as left multiplication by an inverse element, its action commutes with the flow effect of α on it. This is the case because the flow generated by α takes us along edges of the right Cayley graph, and hence manifests as multiplying vertices on the right by each of the generators of A.

Construction of α with Prescribed Centraliser

At last, we are interested in realisability of the entire centraliser of an element of V with a CNP flow graph. By results in Bleak *et al.* in [5] it must be of the form $A \rtimes_\psi \mathbb{Z}$ for a finite group A and $\psi \in \mathrm{Aut}(A)$.

For any choices of a finite group A and automorphism $\psi \in \mathrm{Aut}(A)$ the construction which we will present will result in an element $\alpha_\psi \in V$. This element α_ψ will have the property that $\alpha_\psi^{|\psi|}$ has CNP flow graph, $C_V(\alpha_\psi^{|\psi|}) = \langle \phi(A), \alpha_\psi \rangle$ where ϕ is the embedding of A into V defined in Lemma 4.3.9, and the group $\langle \phi(A), \alpha_\psi \rangle$ naturally decomposes as a semidirect product $\phi(A) \rtimes_\psi \langle \alpha_\psi \rangle$. We will need to show both that $\langle \phi(A), \alpha_\psi \rangle$ centralises $\alpha_\psi^{|\psi|}$, and that any element centralising $\alpha_\psi^{|\psi|}$ is in $\langle \phi(A), \alpha_\psi \rangle$. *This will prove Theorem 4.1.5.*

Construction 4.3.13. Consider a finite group A. We construct a tree pair for α_ψ *almost* like the element α in Construction 4.3.7. Here are the crucial differences:

1. We pick an automorphism $\psi \in \mathrm{Aut}(A)$.
2. We assume that the length of Γ_{h_0} is 1. This means that in step 3 of Construction 4.3.7 the tree T_a has to be chosen appropriately, so that in step 4 the leaf labelled with h_0 must have address 0 or 1.
3. We define α_ψ as follows:

$$\lambda_g \Gamma_{h_0} \overset{\alpha_\psi}{\mapsto} \lambda_{\psi(g)},$$

$$\lambda_g \Gamma_{h_j} \overset{\alpha_\psi}{\mapsto} \lambda'_{\psi(g)h_j} \Gamma'_{h_j},$$

$$\lambda'_g \overset{\alpha_\psi}{\mapsto} \lambda'_{\psi(g)} \Gamma'_{h_0},$$

for all $g \in A$ and for $1 \leq j \leq t$.

If $\psi = \mathrm{id}_A$, then we recover Construction 4.3.7.

Below we will show that unlike in Construction 4.3.7 α_ψ and $\phi(A)$ do not commute unless ψ is trivial.

Lemma 4.3.14. *For any non-trivial $\psi \in \mathrm{Aut}(A)$, α_ψ and $\phi(A)$ do not commute.*

Proof. Let $g, g' \in A$.

$$\big((\lambda_{g'}\Gamma_{h_0})\alpha_\psi\big)\phi(g) = (\lambda_{\psi(g')})\phi(g) = \lambda_{g^{-1}\psi(g')}$$

$$\big((\lambda_{g'}\Gamma_{h_0})\phi(g)\big)\alpha_\psi = (\lambda_{g^{-1}g'}\Gamma_{h_0})\alpha_\psi = \lambda_{\psi(g^{-1}g')}$$

As ψ is not trivial, there exists $g \in A$ s.t. $g \neq \psi(g)$, and for such g:

$$\lambda_{g^{-1}\psi(g')} \neq \lambda_{\psi(g^{-1}g')}$$

□

Definition 4.3.15. Let $\alpha \in V$ be represented by a revealing tree pair (D, R, t). Let the set of all points from $\mathfrak{C}$ underlying periodic neutral leaves of (D, R, t) be denoted $\mathcal{P}_\alpha$. Let the set of all points of $\mathfrak{C}$ lying on finite orbits under the action of α be denoted Per_α. Note that $\mathrm{Per}_\alpha = \mathcal{P}_\alpha \sqcup \mathcal{I}_\alpha$

Before we prove further properties of the centraliser of $\alpha_\psi^{|\psi|}$, we need to prove some more general facts about points in Cantor space under the effect of different powers of the same element:

Lemma 4.3.16. *Let $\alpha \in V$. Then for any positive integer s we have:*

1. $\mathfrak{C}\backslash\mathrm{Per}_\alpha \subseteq \mathfrak{C}\backslash\mathrm{Per}_{\alpha^s}$;
2. $\mathcal{R}_\alpha \subseteq \mathcal{R}_{\alpha^s}$;
3. $\mathcal{A}_\alpha \subseteq \mathcal{A}_{\alpha^s}$;
4. $\mathcal{P}_\alpha \subseteq \mathcal{P}_{\alpha^s}$.

Proof. 1. Consider $p \in \mathfrak{C}\backslash\mathrm{Per}_\alpha$. Then by definition its orbit under the group $\langle\alpha\rangle$ is infinite and given by $\mathcal{O}_p(\langle\alpha\rangle) = \{\ldots, (p)\alpha^{-1}, p, (p)\alpha, \ldots\}$ with $(p)\alpha^i = (p)\alpha^j$ if and only if $i = j$. This orbit under the action of the subgroup of $\langle\alpha\rangle$ generated by α^s splits into s orbits:

$$\mathcal{O}_p(\langle\alpha\rangle) = \mathcal{O}_p(\langle\alpha^s\rangle) \sqcup \mathcal{O}_{(p)\alpha}(\langle\alpha^s\rangle) \sqcup \ldots \mathcal{O}_{(p)\alpha^{s-1}}(\langle\alpha^s\rangle).$$

Each of these orbits consists of infinite number of distinct points. This proves that $\mathfrak{C}\backslash\mathrm{Per}_\alpha \subseteq \mathfrak{C}\backslash\mathrm{Per}_{\alpha^s}$.

2. Consider $p \in \mathcal{R}_\alpha$. Then by definition there is a positive integer k such that $(p)\alpha^k = p$ and there is a prefix λ of p and a finite string Γ such that $(\lambda\Gamma)\alpha^k = \lambda$ and $p = \lambda(\Gamma)^\infty$. Now consider α^s for some positive integer s. We have:

$$(p)(\alpha^s)^k = (p)\underbrace{\alpha^k \ldots \alpha^k}_{s \text{ times}} = (p)\underbrace{\alpha^k \ldots \alpha^k}_{s-1 \text{ times}} = \ldots = p$$
$$(\lambda\Gamma^s)(\alpha^s)^k = (\lambda\Gamma^s)\underbrace{\alpha^k \ldots \alpha^k}_{s \text{ times}} = (\lambda\Gamma^{s-1})\underbrace{\alpha^k \ldots \alpha^k}_{s-1 \text{ times}} = \ldots = \lambda$$

This proves that $p \in \mathcal{R}_{\alpha^s}$.

3. Similarly, consider $p \in \mathcal{A}_\alpha$. Then by definition there is a positive integer k such that $(p)\alpha^k = p$ and there is a prefix λ of p and a finite string Γ such that $(\lambda)\alpha^k = \lambda\Gamma$ and $p = \lambda(\Gamma)^\infty$. Now consider α^s for some positive integer s. We have:

$$(p)(\alpha^s)^k = (p)\underbrace{\alpha^k \ldots \alpha^k}_{s \text{ times}} = (p)\underbrace{\alpha^k \ldots \alpha^k}_{s-1 \text{ times}} = \ldots = p$$
$$(\lambda)(\alpha^s)^k = (\lambda)\underbrace{\alpha^k \ldots \alpha^k}_{s \text{ times}} = (\lambda\Gamma)\underbrace{\alpha^k \ldots \alpha^k}_{s-1 \text{ times}} = \ldots = \lambda\Gamma^s$$

This proves that $p \in \mathcal{A}_{\alpha^s}$.

4. Consider $p \in \mathcal{P}_\alpha$. By definition there is a prefix λ of p such that the orbit of λ under α is given by $\mathcal{O}_\lambda(\langle\alpha\rangle) = \{\lambda, (\lambda)\alpha, \ldots, (\lambda)\alpha^{k-1}\}$ for some positive integer k with $\lambda = (\lambda)\alpha^k$. Now consider what happens to λ under the action of α^s. $\mathcal{O}_\lambda(\langle\alpha^s\rangle) = \{(\lambda)\alpha^s, \ldots, (\lambda)\alpha^{\mathrm{lcm}(k,s)}\}$ with $\lambda = (\lambda)\alpha^{\mathrm{lcm}(k,s)}$. Hence the prefix λ for p is still on a finite orbit, and so in particular $p \in \mathcal{P}_{\alpha^s}$.

□

Corollary 4.3.17. *Each of the containments from Lemma 4.3.16 is in fact an equality.*

Proof. This is because for all $\alpha \in V$ the sets $\mathfrak{C}\backslash\mathrm{Per}_\alpha, \mathcal{R}_\alpha, \mathcal{A}_\alpha$ and $\mathcal{P}_\alpha$ are pairwise disjoint and their union is the whole Cantor set. □

Lemma 4.3.18. *The flow graphs of α_ψ and $\alpha_\psi^{|\psi|}$ are CNP.*

Proof. Let us first consider the flow graph of α_ψ. We will show that there is a sequence of flow lines linking $\lambda_{g_i}(\Gamma_{h_0})^\infty$ to $\lambda'_{g_j}(\Gamma'_{h_0})^\infty$, for any $g_i, g_j \in A$. Note that these are all repelling and attracting points of α. Let $\big(\psi(g_i)\big)^{-1} g_j = h_{i_1} h_{i_2} \ldots h_{i_r}$ for some $h_{i_1}, \ldots, h_{i_r} \in S\backslash\{h_1\}$. Hence:

$$\underbrace{\lambda_{g_i}\Gamma_{h_{i_1}}}_{\lambda_{g_i}} \xrightarrow{\alpha_\psi} \underbrace{\lambda'_{\psi(g_i)h_{i_1}}\Gamma'_{h_{i_1}} \quad \lambda'_{\psi(g_i)h_{i_1}}\Gamma'_{h_1}}_{\lambda'_{\psi(g_i)h_{i_1}}} \xleftarrow{\alpha_\psi}$$

$$\xleftarrow{\alpha_\psi} \underbrace{\lambda_{\psi^{-1}(\psi(g_i)h_{i_1})}\Gamma_{h_1} \quad \lambda_{g_i\psi^{-1}(h_{i_1})}\Gamma_{h_{i_2}}}_{\lambda_{g_i\psi^{-1}(h_{i_1})}} \xrightarrow{\alpha_\psi} \ldots$$

$$\ldots \xrightarrow{\alpha_\psi} \underbrace{\lambda'_{\psi(g_i)h_{i_1}\ldots h_{i_r}}\Gamma'_{h_{i_r}} = \lambda'_{g_j}\Gamma'_{h_{i_r}}}_{\lambda'_{g_j}}$$

Thus α_ψ is connected and as $\mathcal{I}_\alpha \neq \{\}$ it has a CNP flow graph.

Let us now consider the flow graph of $\alpha_\psi^{|\psi|}$. Recall from Corollary 4.3.17 that $\mathcal{R}_{\alpha_\psi} = \mathcal{R}_{\alpha_\psi^{|\psi|}}$ and $\mathcal{A}_{\alpha_\psi} = \mathcal{A}_{\alpha_\psi^{|\psi|}}$. Hence if we show that the flow graph of $\alpha_\psi^{|\psi|}$ is also connected, it will follow that it is CNP. We will show that for each repelling point of $\alpha_\psi^{|\psi|}$ and each attracting point of $\alpha_\psi^{|\psi|}$ there is a sequence of flow lines linking them. More formally, consider

any $g_i, g_j \in A$. Let $(g_i)^{-1}g_j = h_{i_1}h_{i_2}\dots h_{i_r}$ for some non-negative integer r and some $h_{i_1}, \dots, h_{i_r} \in S\backslash\{h_1\}$. Then:

$$\big(\underbrace{\lambda_{g_i}(\Gamma_{h_0})^{|\psi|-1}\Gamma_{h_{i_1}}}_{\lambda_{g_i}}\big)\alpha_\psi^{|\psi|} = \big(\underbrace{\lambda_{\psi(g_i)}(\Gamma_{h_0})^{|\psi|-2}\Gamma_{h_{i_1}}}_{\lambda_{\psi(g_i)}}\big)\alpha_\psi^{|\psi|-1} = \dots$$

$$\dots = \big(\underbrace{\lambda_{\psi^{|\psi|-1}(g_i)}\Gamma_{h_{i_1}}}_{\lambda_{\psi^{|\psi|-1}(g_i)}}\big)\alpha_\psi = \underbrace{\lambda'_{g_ih_{i_1}}\Gamma'_{h_{i_1}}}_{\lambda'_{g_ih_{i_1}}}$$

$$\underbrace{\big(\lambda'_{g_ih_{i_1}}(\Gamma'_{h_0})^{|\psi|-1}\Gamma'_{h_{i_1}}\big)}_{\lambda'_{g_ih_{i_1}}}\alpha_\psi^{-|\psi|} = \underbrace{\big(\lambda'_{\psi^{-1}(g_ih_{i_1})}(\Gamma'_{h_0})^{|\psi|-2}\Gamma'_{h_{i_1}}\big)}_{\lambda'_{\psi^{-1}(g_ih_{i_1})}}\alpha_\psi^{-|\psi|+1} = \dots$$

$$\dots = \underbrace{\big(\lambda'_{\psi^{-|\psi|+1}(g_ih_{i_1})}\Gamma'_{h_1}\big)}_{\lambda'_{\psi^{-|\psi|+1}(g_ih_{i_1})}}\alpha_\psi^{-1} = \underbrace{\lambda_{g_ih_{i_1}}\Gamma_{h_1}}_{\lambda_{g_ih_{i_1}}}$$

$$\underbrace{\big(\lambda_{g_ih_{i_1}}(\Gamma_{h_0})^{|\psi|-1}\Gamma_{h_{i_2}}\big)}_{\lambda_{g_ih_{i_1}}}\alpha_\psi^{|\psi|} = \underbrace{\lambda'_{g_ih_{i_1}h_{i_2}}\Gamma'_{h_{i_2}}}_{\lambda'_{g_ih_{i_1}h_{i_2}}}$$

$$\underbrace{\big(\lambda'_{g_ih_{i_1}h_{i_2}}(\Gamma'_{h_0})^{|\psi|-1}\Gamma'_{h_1}\big)}_{\lambda'_{g_ih_{i_1}h_{i_2}}}\alpha_\psi^{-|\psi|} = \underbrace{\lambda_{g_ih_{i_1}h_{i_2}}\Gamma_{h_1}}_{\lambda_{g_ih_{i_1}h_{i_2}}}$$

$$\vdots$$

$$\underbrace{\big(\lambda_{g_ih_{i_1}\dots h_{i_{r-1}}}(\Gamma_{h_0})^{|\psi|-1}\Gamma_{h_{i_r}}\big)}_{\lambda_{g_ih_{i_1}\dots h_{i_{r-1}}}}\alpha_\psi^{|\psi|} = \underbrace{\lambda'_{g_ih_{i_1}\dots h_{i_r}}\Gamma'_{h_{i_r}}}_{\lambda'_{g_j}}$$

From these calculations we conclude that the flow graph of $\alpha_\psi^{|\psi|}$ is connected, and so by our previous consideration, it is CNP. □

Lemma 4.3.19. *Let A be any finite group, $\psi \in \mathrm{Aut}(A)$ and let $\alpha_\psi \in V$ be the element constructed in Construction 4.3.13. Let ϕ be the embedding defined in Lemma 4.3.9. Then the group $\langle \phi(A), \alpha_\psi\rangle$ decomposes naturally as the semidirect product $\phi(A) \rtimes_\psi \langle\alpha_\psi\rangle$.*

Proof. Let the tree pair constructed in Construction 4.3.13 be (D, R, t). First notice that A is a finite group and α_ψ has infinite order, and so $\phi(A) \cap \langle \alpha_\psi \rangle = \{1_V\}$. We will consider all leaves of the tree R and the effect of $\alpha_\psi^{-1}\phi(g)\alpha_\psi$ on them for any $g \in A$. Our aim is to prove that the effect is the same as that of $\phi(\psi(g))$. Recall that there are three types of leaves to consider:

1. $\lambda_{g'}$
2. $\lambda'_{g'}\Gamma'_{h_j}$ for $h_j \in S$
3. $\lambda'_{g'}\Gamma'_{h_0}$

for all $g' \in A$. Let us then proceed to check:

1.

$$(\lambda_{g'})(\alpha_\psi^{-1}\phi(g)\alpha_\psi) = (\lambda_{\psi^{-1}(g')}\Gamma_{h_0})\phi(g)\alpha_\psi = (\lambda_{g^{-1}\psi^{-1}(g')}\Gamma_{h_0})\alpha_\psi = \lambda_{\psi(g^{-1})g'} = (\lambda_{g'})\phi(\psi(g)).$$

2.

$$(\lambda'_{g'}\Gamma'_{h_j})(\alpha_\psi^{-1}\phi(g)\alpha_\psi) = (\lambda_{\psi^{-1}(g'h_j^{-1})}\Gamma_{h_j})\phi(g)\alpha_\psi = (\lambda_{g^{-1}\psi^{-1}(g'h_j^{-1})}\Gamma_{h_j})\alpha_\psi = \lambda'_{\psi(g^{-1})g'}\Gamma'_{h_j} = (\lambda'_{g'}\Gamma'_{h_j})\phi(\psi(g)).$$

3.

$$(\lambda'_{g'}\Gamma'_{h_0})(\alpha_\psi^{-1}\phi(g)\alpha_\psi) = (\lambda'_{\psi^{-1}(g')})\phi(g)\alpha_\psi = = (\lambda'_{g^{-1}\psi^{-1}(g')})\alpha_\psi = \lambda'_{\psi(g^{-1})g'}\Gamma'_{h_0} = (\lambda'_{g'}\Gamma'_{h_0})\phi(\psi(g)).$$

Hence, $\alpha_\psi^{-1}\phi(g)\alpha_\psi = \phi(\psi(g))$ and so the group $\langle \phi(A), \alpha_\psi \rangle$ decomposes naturally as the semidirect product $\phi(A) \rtimes_\psi \langle \alpha_\psi \rangle$. □

We will use the lemma above to prove the following:

Lemma 4.3.20. *Let A be any finite group, $\psi \in \mathrm{Aut}(A)$ and let $\alpha_\psi \in V$ be the element constructed in Construction 4.3.13 from A. Let ϕ be the embedding defined in Lemma 4.3.9. Then $\phi(A)$ centralises $\alpha_\psi^{|\psi|}$. Moreover, $\phi(A)$ is the torsion subgroup of $C_V(\alpha_\psi^{|\psi|})$.*

Proof. Consider $g \in A$. By Lemma 4.3.19 we have $\alpha_\psi^{-1}\phi(g)\alpha_\psi = \phi(\psi(g))$. Hence $\alpha_\psi^{-|\psi|}\phi(g)\alpha_\psi^{|\psi|} = \phi(\psi^{|\psi|}(g)) = \phi(g)$. This implies that $\phi(g)$ and $\alpha_\psi^{|\psi|}$ commute and so $\phi(g) \in C_V(\alpha_\psi^{|\psi|})$. Hence $\phi(A) \subseteq C_V(\alpha_\psi^{|\psi|})$. As $\phi(A)$ is a finite group and $|\phi(A)| = |\mathcal{R}_{\alpha_\psi^{|\psi|}}|$, by Remark 4.3.5 the group $\phi(A)$ is the torsion subgroup of the centraliser $C_V(\alpha_\psi^{|\psi|})$. □

Lemma 4.3.21. *Consider α_ψ constructed from a finite group A in Construction 4.3.13. The centraliser $C_V(\alpha_\psi^{|\psi|})$ decomposes naturally as*

$$\phi(A) \rtimes_\psi \langle \alpha_\psi \rangle.$$

This proves our Theorem 4.1.5 for $\alpha = \alpha_\psi^{|\psi|}$.

Proof. By Lemma 4.3.20 the group $\phi(A)$ is the full torsion of the centraliser $C_V(\alpha_\psi^{|\psi|})$. Thus by Lemma 4.2.16 the centraliser is of the form $\phi(A) \rtimes \mathbb{Z}$ with the generator of $\mathbb{Z}$ commuting with $\alpha_\psi^{|\psi|}$. We know that α_ψ commutes with $\alpha_\psi^{|\psi|}$ and so the group $\langle \phi(A), \alpha_\psi \rangle$ is a subgroup of $C_V(\alpha_\psi^{|\psi|})$. Recall from Lemma 4.3.19 that the group $\langle \phi(A), \alpha_\psi \rangle$ decomposes naturally as a semidirect product $\phi(A) \rtimes_\psi \langle \alpha_\psi \rangle$. Now we will show that for all $\beta \in C_V(\alpha_\psi^{|\psi|})$ we have $\beta \in \phi(A) \rtimes_\psi \langle \alpha_\psi \rangle$. This will imply that $\phi(A) \rtimes_\psi \langle \alpha_\psi \rangle = C_V(\alpha_\psi^{|\psi|})$.

Hence, consider any $\beta \in C_V(\alpha_\psi^{|\psi|})$. By Corollary 4.2.5 we know that β permutes elements of the set $\mathcal{R}_{\alpha_\psi^{|\psi|}}$. Let r_e be the repelling point $\lambda_e(\Gamma_{h_0})^\infty$. Note that r_e is a stationary point of α_ψ as $(\lambda_e\Gamma_{h_0})\alpha_\psi = \lambda_{\psi(e)} = \lambda_e$. Let $r_g = (r_e)\beta$ for some $g \in A$. By our construction, there is a unique element of $\phi(A)$ which takes r_g to r_e. Notice that $(r_g)\phi(g) = r_{g^{-1}g} = r_e$. Therefore, $(r_e)(\beta\phi(g)) = (r_g)\phi(g) = r_e$ and so $\beta\phi(g)$ stabilises r_e. Let $x = f_{r_e}(\beta\phi(g))$ and note that $x \in \mathbb{Z}$. *Now recall from Construction 4.3.13 that we assumed that* $|\Gamma_{h_0}| = 1$. As r_e is a stationary point of α_ψ, we have $f_{r_e}(\alpha_\psi) = L(\lambda_e\Gamma_{h_0}) - L(\lambda_e) = L(\Gamma_{h_0}) = 1$. Thus by Lemma 4.2.12 we have:

$$f_{r_e}(\beta\phi(g)\alpha_\psi^{-x}) = f_{r_e}(\beta\phi(g)) + f_{(r_e)\beta\phi(g)}(\alpha_\psi^{-x}) = x + f_{r_e}(\alpha_\psi^{-x}).$$

But as $(r_e)\alpha_\psi = r_e$ the restriction map $f_{r_e}|_{\langle\alpha_\psi\rangle}$ is a group homomorphism and so $f_{r_e}(\alpha_\psi^{-x}) = -x \cdot f_{r_e}(\alpha_\psi) = -x$. This implies that:

$$f_{r_e}(\beta\phi(g)\alpha_\psi^{-x}) = x - x = 0.$$

To summarise $\beta\phi(g)\alpha_\psi^{-x}$ and $\phi(e)$ both stabilise the repelling point r_e and $f_{r_e}(\beta\phi(g)\alpha_\psi^{-x}) = 0 = f_{r_e}(\phi(e))$. Thus by Proposition 4.2.18 we have $\beta\phi(g)\alpha_\psi^{-x} = \phi(e)$ which is equivalent to $\beta = \alpha_\psi^x\phi(g^{-1})$. This implies that $\beta \in \langle \phi(A), \alpha_\psi \rangle$ as required. □

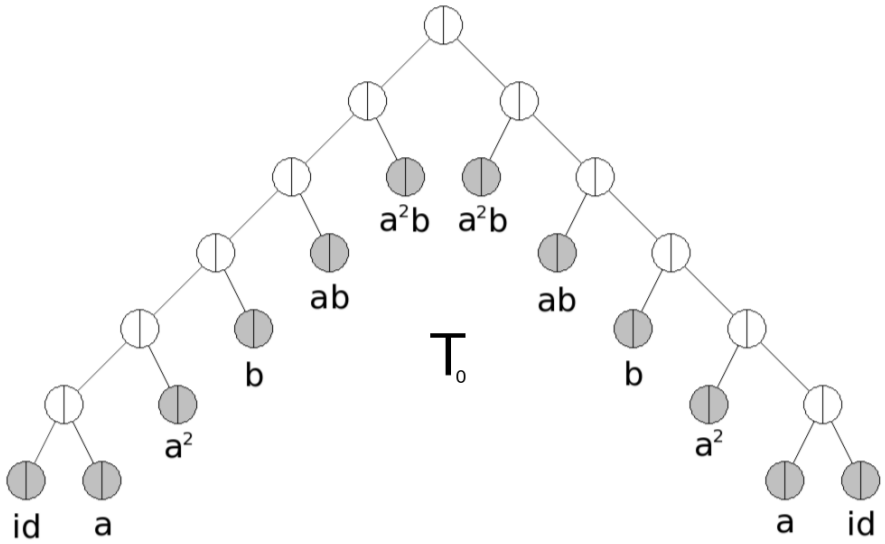
Examples

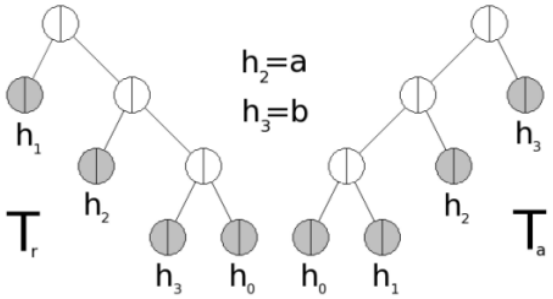
Finally we are ready to give justified examples of elements of V with prescribed centralisers. In Example 4.3.22 below the torsion subgroup of the centraliser is isomorphic to S_3. In Example 4.3.23 below the centraliser is isomorphic to $C_3 \rtimes_\psi \mathbb{Z}$ where ψ is the non–trivial automorphism of C_3. The choice of examples was made so that Example 4.3.22 presents the smallest case where the torsion subgroup of the centraliser is a non–abelian group and Example 4.3.23 presents the smallest case where the automorphism ψ is non–trivial.

Example 4.3.22. Consider $A = S_3$ with the presentation

$$S_3 = \langle a, b \mid a^3 = 1 = b^2, ba = a^2b \rangle = \{e, a, a^2, b, ab, a^2b\}$$

plugged into Construction 4.3.7. Here we will exhibit specific choices regarding the trees T_0, T_r and T_a together with labeling of their leaves by elements of S_3 for the generating set $\{a, b\}$ of S_3. Note that the copies of the tree T_r are going to be rooted at the leaves with prefix 0 of the tree T_0 (the left part of the tree) and the copies of the tree T_a are going to be rooted at the leaves with prefix 1 of the tree T_0 (the right part of the tree).

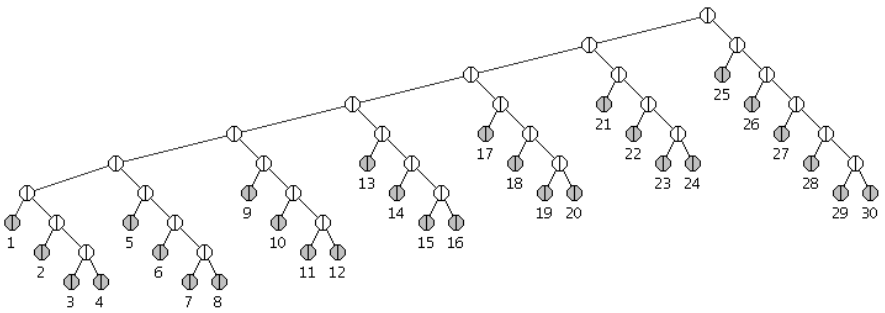




Hence we can read off the following information:

	$\lambda_e = 000000$	$\lambda'_e = 111111$	
$\Gamma_{h_0} = 111$	$\lambda_a = 000001$	$\lambda'_a = 111110$	$\Gamma'_{h_0} = 000$
$\Gamma_e = 0$	$\lambda_{a^2} = 00001$	$\lambda'_{a^2} = 11110$	$\Gamma'_e = 001$
$\Gamma_a = 10$	$\lambda_b = 0001$	$\lambda'_b = 1110$	$\Gamma'_a = 01$
$\Gamma_b = 110$	$\lambda_{ab} = 001$	$\lambda'_{ab} = 110$	$\Gamma'_b = 1$
	$\lambda_{a^2b} = 01$	$\lambda'_{a^2b} = 10$	

Given the choices of T_0 and T_r, the domain tree D from the tree pair (D, R, t) which we are constructing for α is given by:



Recall that α represents the following prefix substitutions:

$$\begin{aligned}(\lambda_g\Gamma_{h_0})\alpha &= \lambda_g \\ (\lambda_g\Gamma_h)\alpha &= \lambda'_{gh}\Gamma'_h \quad \text{for } h \neq h_0 \\ (\lambda'_g)\alpha &= \lambda'_g\Gamma'_{h_0}\end{aligned}$$

Hence we can compute where each of the leaves of the domain tree D are mapped by α:

$$\begin{aligned}1 : (\lambda_e\Gamma_e)\alpha &= \lambda'_e\Gamma'_e \qquad & 2 : (\lambda_e\Gamma_a)\alpha &= \lambda'_a\Gamma'_a \\ 3 : (\lambda_e\Gamma_b)\alpha &= \lambda'_b\Gamma'_b \qquad & 4 : (\lambda_e\Gamma_{h_0})\alpha &= \lambda_e\end{aligned}$$

$$\begin{aligned}5 : (\lambda_a\Gamma_e)\alpha &= \lambda'_a\Gamma'_e \qquad & 6 : (\lambda_a\Gamma_a)\alpha &= \lambda'_{a^2}\Gamma'_a \\ 7 : (\lambda_a\Gamma_b)\alpha &= \lambda'_{ab}\Gamma'_b \qquad & 8 : (\lambda_a\Gamma_{h_0})\alpha &= \lambda_a\end{aligned}$$

$$\begin{aligned}9 : (\lambda_{a^2}\Gamma_e)\alpha &= \lambda'_{a^2}\Gamma'_e \qquad & 10 : (\lambda_{a^2}\Gamma_a)\alpha &= \lambda'_e\Gamma'_a \\ 11 : (\lambda_{a^2}\Gamma_b)\alpha &= \lambda'_{a^2b}\Gamma'_b \qquad & 12 : (\lambda_{a^2}\Gamma_{h_0})\alpha &= \lambda_{a^2}\end{aligned}$$

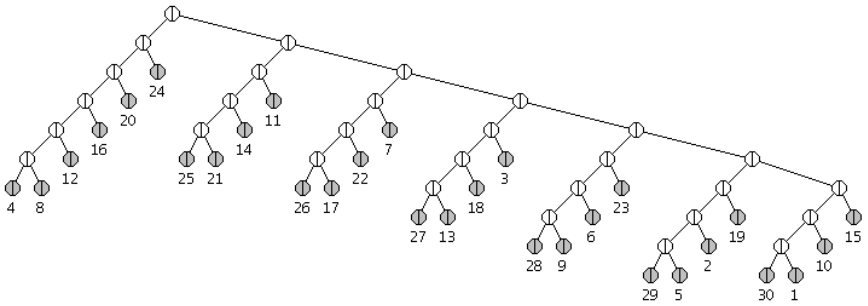
$$13 : (\lambda_b\Gamma_e)\alpha = \lambda'_b\Gamma'_e \qquad 14 : (\lambda_b\Gamma_a)\alpha = \lambda'_{a^2b}\Gamma'_a$$
$$15 : (\lambda_b\Gamma_b)\alpha = \lambda'_e\Gamma'_b \qquad 16 : (\lambda_b\Gamma_{h_0})\alpha = \lambda_b$$

$$17 : (\lambda_{ab}\Gamma_e)\alpha = \lambda'_{ab}\Gamma'_e \qquad 18 : (\lambda_{ab}\Gamma_a)\alpha = \lambda'_b\Gamma'_a$$
$$19 : (\lambda_{ab}\Gamma_b)\alpha = \lambda'_a\Gamma'_b \qquad 20 : (\lambda_{ab}\Gamma_{h_0})\alpha = \lambda_{ab}$$

$$21 : (\lambda_{a^2b}\Gamma_e)\alpha = \lambda'_{a^2b}\Gamma'_e \qquad 22 : (\lambda_{a^2b}\Gamma_a)\alpha = \lambda'_{ab}\Gamma'_a$$
$$23 : (\lambda_{a^2b}\Gamma_b)\alpha = \lambda'_{a^2}\Gamma'_b \qquad 24 : (\lambda_{a^2b}\Gamma_{h_0})\alpha = \lambda_{a^2b}$$

$$25 : (\lambda'_{a^2b})\alpha = \lambda'_{a^2b}\Gamma'_{h_0} \qquad 26 : (\lambda'_{ab})\alpha = \lambda'_{ab}\Gamma'_{h_0}$$
$$27 : (\lambda'_b)\alpha = \lambda'_b\Gamma'_{h_0} \qquad 28 : (\lambda'_{a^2})\alpha = \lambda'_{a^2}\Gamma'_{h_0}$$
$$29 : (\lambda'_a)\alpha = \lambda'_a\Gamma'_{h_0} \qquad 30 : (\lambda'_e)\alpha = \lambda'_e\Gamma'_{h_0}$$

Therefore the range tree R from the tree pair (D, R, t) representing our α is given by:



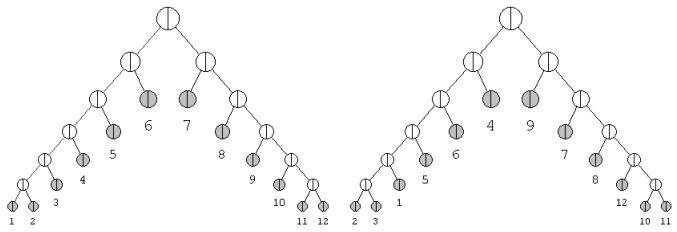
The torsion subgroup of the centraliser of α is the group $\phi(S_3)$, for ϕ defined in Lemma 4.3.9, which is generated by two elements of V, namely $\phi(a)$ and $\phi(b)$. We will analyse the effect each of them has on leaves of the tree T_0 and hence read off the tree pairs for them. The general rules of an action by $\phi(g)$ are given by:

$$(\lambda_{g'})\phi(g) = \lambda_{g^{-1}g'}$$
$$(\lambda'_{g'})\phi(g) = \lambda'_{g^{-1}g'}$$

Hence the effect of $\phi(a)$ on the leaves of T_0 is:

$$\begin{aligned}
(\lambda_e)\phi(a) = \lambda_{a^{-1}e} = \lambda_{a^2} \quad & (\lambda'_e)\phi(a) = \lambda'_{a^2} \\
(\lambda_a)\phi(a) = \lambda_{a^{-1}a} = \lambda_e \quad & (\lambda'_a)\phi(a) = \lambda'_e \\
(\lambda_{a^2})\phi(a) = \lambda_{a^{-1}a^2} = \lambda_a \quad & (\lambda'_{a^2})\phi(a) = \lambda'_a \\
(\lambda_b)\phi(a) = \lambda_{a^{-1}b} = \lambda_{a^2b} \quad & (\lambda'_b)\phi(a) = \lambda'_{a^2b} \\
(\lambda_{ab})\phi(a) = \lambda_{a^{-1}ab} = \lambda_b \quad & (\lambda'_{ab})\phi(a) = \lambda'_b \\
(\lambda_{a^2b})\phi(a) = \lambda_{a^{-1}a^2b} = \lambda_{ab} \quad & (\lambda'_{a^2b})\phi(a) = \lambda'_{ab}
\end{aligned}$$

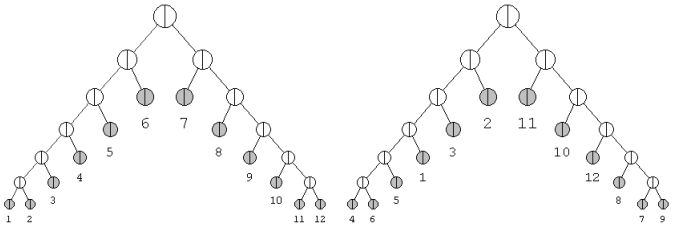
Therefore the tree pair corresponding to $\phi(a)$ is given by:



Similarly, the effect of $\phi(b)$ on the leaves of T_0 is:

$$\begin{aligned}
(\lambda_e)\phi(b) = \lambda_{b^{-1}e} = \lambda_b \quad & (\lambda'_e)\phi(b) = \lambda'_b \\
(\lambda_a)\phi(b) = \lambda_{b^{-1}a} = \lambda_{a^2b} \quad & (\lambda'_a)\phi(b) = \lambda'_{a^2b} \\
(\lambda_{a^2})\phi(b) = \lambda_{b^{-1}a^2} = \lambda_{ab} \quad & (\lambda'_{a^2})\phi(b) = \lambda'_{ab} \\
(\lambda_b)\phi(b) = \lambda_{b^{-1}b} = \lambda_e \quad & (\lambda'_b)\phi(b) = \lambda'_e \\
(\lambda_{ab})\phi(b) = \lambda_{b^{-1}ab} = \lambda_{a^2} \quad & (\lambda'_{ab})\phi(b) = \lambda'_{a^2} \\
(\lambda_{a^2b})\phi(b) = \lambda_{b^{-1}a^2b} = \lambda_a \quad & (\lambda'_{a^2b})\phi(b) = \lambda'_a
\end{aligned}$$

Therefore the tree pair corresponding to $\phi(b)$ is given by:



We conclude that $\langle \phi(a), \phi(b) \rangle$ is the group which corresponds to the torsion subgroup of centraliser of α in V.

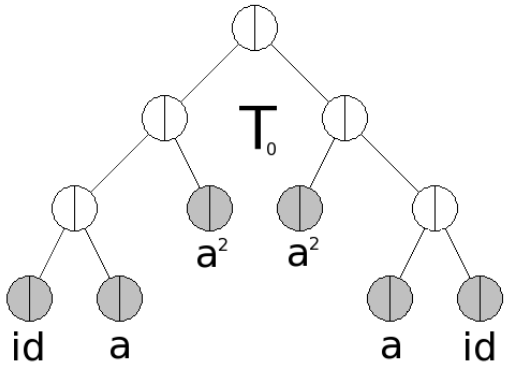
The smallest non-commutative group has been chosen as an example for A as an early problem was to decide whether torsion of the centraliser could be realised only by abelian groups (see Question (2) of [5]).

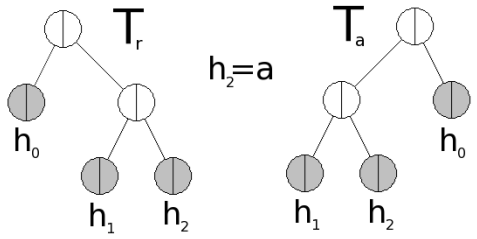
Example 4.3.23. We will construct a tree pair for α_ψ such that the centraliser of $\alpha_\psi^{|\psi|}$ is of the following form:

$$C_3 \rtimes_\psi \mathbb{Z} \cong \langle a, b \mid a^3 = 1, a^b = a^2 \rangle$$

and where ψ is the non-trivial automorphism of C_3.

Consider $A = C_3$ with the presentation $C_3 = \langle a \mid a^3 = 1 \rangle$ and ψ such that $\psi(a) = a^2$ plugged into Construction 4.3.13. Here we will exhibit specific choices regarding the trees T_0, T_r and T_a together with labeling of their leaves by elements of C_3 for the generating set $\{a\}$ of C_3. Note that the copies of the tree T_r are going to be rooted at the leaves with prefix 0 of the tree T_0 (the left part of the tree) and the copies of the tree T_a are going to be rooted at the leaves with prefix 1 of the tree T_0 (the right part of the tree).





Hence we can read off the following information:

$$\begin{array}{llll} \Gamma_{h_0} = 0 & \lambda_e = 000 & \lambda'_e = 111 & \Gamma'_{h_0} = 1 \\ \Gamma_e = 10 & \lambda_a = 001 & \lambda'_a = 110 & \Gamma'_e = 00 \\ \Gamma_a = 11 & \lambda_{a^2} = 01 & \lambda'_{a^2} = 10 & \Gamma'_a = 01 \end{array}$$

Recall that α_ψ represents the following prefix substitutions:

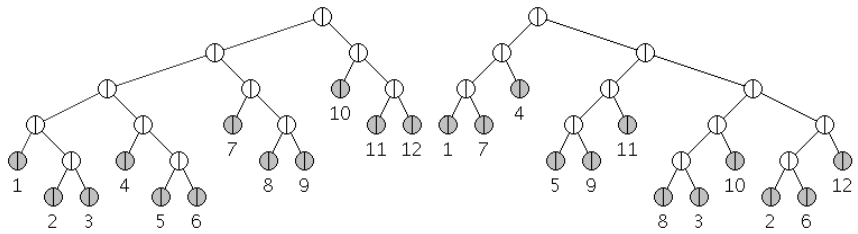
$$\begin{aligned} (\lambda_g \Gamma_{h_0})\alpha_\psi &= \lambda_{\psi(g)} \\ (\lambda_g \Gamma_h)\alpha_\psi &= \lambda'_{\psi(g)h}\Gamma'_h \quad \text{for } h \neq h_0 \\ (\lambda'_g)\alpha_\psi &= \lambda'_{\psi(g)}\Gamma'_{h_0} \end{aligned}$$

Thus we can compute more specifically:

$$\begin{array}{ll} 1 : (\lambda_e \Gamma_{h_0})\alpha_\psi = \lambda_e & 4 : (\lambda_a \Gamma_{h_0})\alpha_\psi = \lambda_{a^2} \\ 2 : (\lambda_e \Gamma_e)\alpha_\psi = \lambda'_e \Gamma'_e & 5 : (\lambda_a \Gamma_e)\alpha_\psi = \lambda'_{a^2}\Gamma'_e \\ 3 : (\lambda_e \Gamma_a)\alpha_\psi = \lambda'_a \Gamma'_a & 6 : (\lambda_a \Gamma_a)\alpha_\psi = \lambda'_e \Gamma'_a \end{array}$$

$$\begin{array}{ll} 7 : (\lambda_{a^2} \Gamma_{h_0})\alpha_\psi = \lambda_a & 10 : (\lambda'_{a^2})\alpha_\psi = \lambda'_a \Gamma'_{h_0} \\ 8 : (\lambda_{a^2} \Gamma_e)\alpha_\psi = \lambda'_a \Gamma'_e & 11 : (\lambda'_a)\alpha_\psi = \lambda'_{a^2}\Gamma'_{h_0} \\ 9 : (\lambda_{a^2} \Gamma_a)\alpha_\psi = \lambda'_{a^2}\Gamma'_a & 12 : (\lambda'_e)\alpha_\psi = \lambda'_e \Gamma'_{h_0} \end{array}$$

Therefore the tree pair (D, R, t) representing our α_ψ is given by:



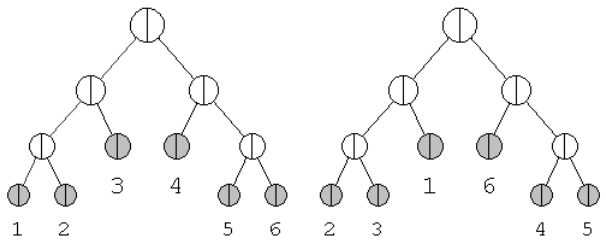
The torsion subgroup of the centraliser of $\alpha_\psi^{|\psi|}$ is the group $\phi(C_3)$ for ϕ defined in Lemma 4.3.9, which is generated by one element of V, namely $\phi(a)$. We will analyse what effect it has on leaves of the tree T_0 and hence read off the tree pair for it. The general rules of an action by $\phi(g)$ are given by:

$$(\lambda_{g'})\phi(g) = \lambda_{g^{-1}g'}$$
$$(\lambda'_{g'})\phi(g) = \lambda'_{g^{-1}g'}$$

Hence the effect of $\phi(a)$ on the leaves of T_0 is:

$$\begin{aligned} (\lambda_e)\phi(a) = \lambda_{a^{-1}e} = \lambda_{a^2} \quad & (\lambda'_e)\phi(a) = \lambda'_{a^2} \\ (\lambda_a)\phi(a) = \lambda_{a^{-1}a} = \lambda_e \quad & (\lambda'_a)\phi(a) = \lambda'_e \\ (\lambda_{a^2})\phi(a) = \lambda_{a^{-1}a^2} = \lambda_a \quad & (\lambda'_{a^2})\phi(a) = \lambda'_a \end{aligned}$$

Therefore the tree pair corresponding to $\phi(a)$ is given by:



We conclude that $\langle \phi(a) \rangle$ is the torsion subgroup of centraliser of $\alpha_\psi^{|\psi|}$ in V and that the centraliser $C_V(\alpha_\psi^{|\psi|})$ naturally decomposes as a semidirect product $\phi(C_3) \rtimes_\psi \langle \alpha_\psi \rangle$.

Chapter 5

Free Products

In this chapter we present my contribution to joint work with Collin Bleak and Francesco Matucci [4]. We intend to familiarise the reader with the dynamics occurring on Cantor space under the action of a subgroup of V when that group decomposes as a non–trivial free product. The new work presented in this chapter relates to the work of Bleak and Salazar–Díaz on free products in V in [8] where the non–embedding of $\mathbb{Z}^2 * \mathbb{Z}$ into V was proven.

The original question which inspired this work was posed by Bleak and Salazar–Díaz in [8]. They asked whether each subgroup of V which can be expressed as a free product of two non–trivial groups admits a Ping–Pong action on Cantor space. A Ping–Pong, as in Definition 5.2.1, is a dynamical structure occurring on a set on which a group acts which assures decomposition of this group as a free product. The converse is not true in general, as there are known actions of free groups on spaces which admit no Ping–Pong structure, as shown by White [32].

We show in this chapter that whenever we have a subgroup of V which can be decomposed as a free product of two groups which are not both torsion, there is a Ping–Pong action on a set of subsets of Cantor space. The Ping–Pong sets are constructed using important points, introduced in Section 3.4, of an infinite order element of one of these groups. We continue to work towards expanding our findings, especially to search for Ping–Pong dynamics reflected in types of sets distinct to the ones from our construction.

We also want to draw attention to the results of Bennett and Bleak presented in [2] which state that the class of demonstrable subgroups of V is precisely the class of virtually free subgroups of V. Note that

each free product of two finite groups is a virtually free group. For a subgroup of V, being demonstrable means that there is a demonstrative embedding of it into V. For a subgroup of V to be demonstrative means that it acts on $\mathfrak{C}$ in a specific way. Particularly, there is a finite address $w \in \{0,1\}^*$ such that, for each non–trivial element α of the group, there is no point in Cantor space underlying addresses w and $w\alpha$ simultaneously. This implies that if a demonstrative group can be decomposed as a free product, it admits a Ping–Pong action on $\mathfrak{C}$. It is worth noting that this Ping–Pong construction uses a different type of subsets of $\mathfrak{C}$ than the one described in our work.

5.1 Statement of Results

The main result of this chapter is given by the following theorem:

Theorem 5.1.1. *Consider a group generated by two subgroups of V which are not both torsion, say A and B. Suppose that the groups $A * B$ and $\langle A, B\rangle$ are isomorphic via the canonical map:*

$$f : A * B \longrightarrow \langle A, B\rangle$$
$$a_1 \cdot b_1 \cdot a_2 \cdot b_2 \cdot \ldots \cdot a_k \cdot b_k \longmapsto a_1 b_1 a_2 b_2 \ldots a_k b_k$$

for $k \in \mathbb{Z}$, $a_1 \in A$, $a_2, \ldots, a_k \in A\backslash\{e\}$, $b_1, \ldots, b_{k-1} \in B\backslash\{e\}$ and $b_k \in B$, i.e. for any element of $A * B$ in its normal form. Then there is a set X of subsets of $\mathfrak{C}$ such that the induced action of $\langle A, B\rangle$ is a Ping–Pong action.

We will say that $\langle A, B\rangle$ *canonically decomposes* or *factors as a free product* if the map f defined above is an isomorphism. We will further explain what it means to admit a Ping–Pong action in Definition 5.2.1.

5.2 Literature Review

In this section, we present the statement and proof of Ping–Pong Theorem, which is based on the proof found in de la Harpe [16]. The original result, however, was first used by Klein in [15], [20] at the end of 19th century. We also restate a theorem from Bleak and Salazar–Díaz [8] which will be crucial for reaching conclusions of this chapter.

Ping–Pong

Definition 5.2.1 (Ping–Pong)**.** Consider a group G acting on a set X. Suppose A, B are non–trivial subgroups of G, such that $|A| > 2$. Suppose that there are subsets $X_A, X_B \subset X$, such that:

1. $X_B \backslash X_A \neq \emptyset$;
2. for all $a \in A \backslash \{e\}$, we have $(X_B)a \subseteq X_A$;
3. for all $b \in B \backslash \{e\}$, we have $(X_A)b \subseteq X_B$.

Then we say that *the groups A and B admit a Ping–Pong on the set X*, or *on the set $X_A \cup X_B$*, or *on the sets X_A and X_B*.

We can also say that *the groups A and B admit Ping–Pong dynamics* or a *Ping–Pong action.*

Theorem 5.2.2 (Ping–Pong Theorem)**.** *Suppose that G, X, A, B, X_A and X_B are as presented in Definition 5.2.1. Then $\langle A, B \rangle$ factors as free product $A * B$.*

Proof. Suppose that the groups A, B and the sets X_A, X_B are as in Definition 5.2.1 and suppose that the Conditions 1), 2) and 3) from Definition 5.2.1 hold.

Let $c_1 \cdot c_2 \cdot \ldots \cdot c_k$ for some positive integer k be a non–trivial expression in a normal form of a formal free product $A * B$. Namely, $c_1, \ldots, c_k$ are all non–trivial and for any consecutive c_i and c_{i+1} (for $1 \leq i < k$), c_i and c_{i+1} are in distinct groups A or B.

Since $|A| > 2$, there is $a \in A$ such that $a^{-1}(c_1 \cdot \ldots \cdot c_k)a$ resolves as a product $d_1 \cdot d_2 \cdot \ldots \cdot d_m$ in the normal form of $A * B$ such that $d_1, d_m \in A$. Consider the following cases:

$$\begin{aligned} \text{If } c_1, c_k \in A \qquad & \text{then chose } a = e. \\ \text{If } c_1, c_k \in B \qquad & \text{then chose } a \in A \setminus \{e\}. \\ \text{If } c_1 \in A, c_k \in B \qquad & \text{then chose } a \in A \setminus \{e, c_1\}. \\ \text{If } c_1 \in B, c_k \in A \qquad & \text{then chose } a \in A \setminus \{e, c_k^{-1}\}. \end{aligned}$$

In all cases the resolved product $a^{-1}c_1 \cdot \ldots \cdot c_k a = d_1 \cdot d_2 \cdot \ldots \cdot d_m$ begins and ends with a non–trivial elements of A. Now consider $x \in X_B \setminus X_A$,

which exists by Condition 1). Then $(x)d_1 \dots d_m = y \in X_A$ by Conditions 2) and 3), and so $x \neq y$. Therefore, the product $d_1 \dots d_m$ is non–trivial. As $d_1 \dots d_m = (c_1 \dots c_k)^a$, we also conclude that $c_1 \dots c_k$ is a non–trivial word. As $c_1 \cdot \dots \cdot c_k$ was chosen as arbitrary and non–trivial in the normal form of $A * B$, this proves that there is no non–empty reduced word which reduces to identity in $\langle A, B \rangle$. This proves the theorem, that A and B span a free product. □

Dynamics of Free Products

First, we will prove Lemma 5.2.3 which we will need for Theorem 5.2.4. The lemma is new as stated and it uses prefix substitution, as consistent with this thesis. A non–algebraic proof was presented in Bleak and Salazar–Díaz [8].

Lemma 5.2.3. *Let $a, b \in V$ be non–torsion elements. Suppose that $\mathcal{I}_a \subseteq \mathcal{I}_b$. Let m and n be the smallest positive integers such that $c = a^m$ and $d = b^n$ have no non–trivial periodic orbits on Cantor space (see Remark 3.4.20). Then the commutator $[c, d]$ fixes some prefix of each important point of a.*

Proof. First, notice that $\mathcal{I}_c = \mathcal{I}_{a^m} = \mathcal{I}_a \subseteq \mathcal{I}_b = \mathcal{I}_{b^n} = \mathcal{I}_d$ by Corollary 4.3.17. Note that by definition of c, all important points of c are stationary under application of c. Similarly, by definition of d, all important points of d are stationary under application of d. Then suppose that $p \in \mathcal{I}_c = \mathcal{I}_c \cap \mathcal{I}_d$. This means that $(p)c = p = (p)d$. Moreover, there are finite words $\lambda_c, \lambda_d, \Delta_c, \Delta_d$ such that $\lambda_c(\Delta_c)^\infty = p = \lambda_d(\Delta_d)^\infty$ and one of the following holds:

1. We have $(\lambda_c\Delta_c)c = \lambda_c$ and $(\lambda_d\Delta_d)d = \lambda_d$. This means that $p \in \mathcal{R}_c \cap \mathcal{R}_d$.
2. We have $(\lambda_c)c = \lambda_c\Delta_c$ and $(\lambda_d\Delta_d)d = \lambda_d$. This means that $p \in \mathcal{A}_c \cap \mathcal{R}_d$.
3. We have $(\lambda_c\Delta_c)c = \lambda_c$ and $(\lambda_d)d = \lambda_d\Delta_d$. This means that $p \in \mathcal{R}_c \cap \mathcal{A}_d$.
4. We have $(\lambda_c)c = \lambda_c\Delta_c$ and $(\lambda_d)d = \lambda_d\Delta_d$. This means that $p \in \mathcal{A}_c \cap \mathcal{A}_d$.

Now, let c' be defined as follows:

- if $p \in \mathcal{R}_c$ then $c' = c$;
- if $p \in \mathcal{A}_c$ then $c' = c^{-1}$.

Similarly, let d' be defined as follows:

- if $p \in \mathcal{R}_d$ then $d' = d$;
- if $p \in \mathcal{A}_d$ then $d' = d^{-1}$.

Then we can conclude that $(\lambda_c\Delta_c)c' = \lambda_c$ and $(\lambda_d\Delta_d)d' = \lambda_d$. Without loss of generality, suppose that $|\lambda_c| \geq |\lambda_d|$. As λ_c and λ_d are both prefixes for the point p, there is a (possibly empty) finite word λ' such that $\lambda_c = \lambda_d\lambda'$. Note though that λ_c is a prefix of $p = \lambda_d(\Delta_d)^\infty$. Hence we have $\lambda' = (\Delta_d)^k b_1 b_2 \dots b_l$ for some non–negative integer k, $b_i \in \{0, 1\}$ for all $1 \leq i \leq l$, and for $b_1 b_2 \dots b_l$ a proper, possibly empty prefix of Δ_d. Say that $\Delta_d = b_1 b_2 \dots b_l b_{l+1} \dots b_{|\Delta_d|}$. Then:

$$(\lambda_c b_{l+1} \dots b_{|\Delta_d|} b_1 b_2 \dots b_l)d' = (\lambda_d(\Delta_d)^k b_1 b_2 \dots b_l b_{l+1} \dots b_{|\Delta_d|} b_1 b_2 \dots b_l)d' = (\lambda_d(\Delta_d)^{k+1} b_1 b_2 \dots b_l)d' = \lambda_d(\Delta_d)^k b_1 b_2 \dots b_l = \lambda_c$$

Let us then define $\Gamma_d = b_{l+1} \dots b_{|\Delta_d|} b_1 b_2 \dots b_l$ and $\lambda = \lambda_c$. Thus $(\lambda\Gamma_d)d' = \lambda$ and so $p = \lambda(\Gamma_d)^\infty$. Now observe that:

$$p = (p)d'^{-1} = (\lambda(\Delta_c)^\infty)d'^{-1} = \lambda\Gamma_d(\Delta_c)^\infty$$
$$p = (p)c'^{-1} = (\lambda(\Gamma_d)^\infty)c'^{-1} = \lambda\Delta_c(\Gamma_d)^\infty$$

Therefore, in particular we have $\lambda\Gamma_d\Delta_c = \lambda\Delta_c\Gamma_d$, as both the addresses have the same length and are both a prefix for p. We claim that the commutator $[c, d]$ always fixes the address $\lambda\Gamma_d\Delta_c$.

We have the following cases to consider:

1. We have $(\lambda\Delta_c)c = \lambda$ and $(\lambda\Gamma_d)d = \lambda$.

$$(\lambda\Gamma_d\Delta_c)[c, d] = (\lambda\Gamma_d\Delta_c)c^{-1}d^{-1}cd = (((\lambda)c^{-1})\Gamma_d\Delta_c)d^{-1}cd = (\lambda\Delta_c\Gamma_d\Delta_c)d^{-1}cd = (((\lambda)d^{-1})\Delta_c\Gamma_d\Delta_c)cd = (\lambda\Gamma_d\Delta_c\Gamma_d\Delta_c)cd = ((\lambda\Gamma_d\Delta_c)\Gamma_d\Delta_c)cd = ((\lambda\Delta_c\Gamma_d)\Gamma_d\Delta_c)cd = (((\lambda\Delta_c)c)\Gamma_d\Gamma_d\Delta_c)d = (\lambda\Gamma_d\Gamma_d\Delta_c)d = ((\lambda\Gamma_d)d)\Gamma_d\Delta_c = \lambda\Gamma_d\Delta_c$$

2. We have $(\lambda)c = \lambda\Delta_c$ and $(\lambda\Gamma_d)d = \lambda$.

$$(\lambda\Gamma_d\Delta_c)[c,d] = (\lambda\Gamma_d\Delta_c)c^{-1}d^{-1}cd = (\lambda\Delta_c\Gamma_d)c^{-1}d^{-1}cd = (\lambda\Gamma_d)d^{-1}cd = (\lambda\Gamma_d\Gamma_d)cd = (\lambda\Delta_c\Gamma_d\Gamma_d)d = (\lambda\Gamma_d\Delta_c\Gamma_d)d = \lambda\Delta_c\Gamma_d = \lambda\Gamma_d\Delta_c$$

3. We have $(\lambda\Delta_c)c = \lambda$ and $(\lambda)d = \lambda_d\Gamma_d$.

$$(\lambda\Gamma_d\Delta_c)[c,d] = (\lambda\Gamma_d\Delta_c)c^{-1}d^{-1}cd = (\lambda\Delta_c\Gamma_d\Delta_c)d^{-1}cd = (\lambda\Gamma_d\Delta_c\Delta_c)d^{-1}cd = (\lambda\Delta_c\Delta_c)cd = (\lambda\Delta_c)d = \lambda\Gamma_d\Delta_c$$

4. We have $(\lambda)c = \lambda\Delta_c$ and $(\lambda)d = \lambda_d\Gamma_d$.

$$(\lambda\Gamma_d\Delta_c)[c,d] = (\lambda\Gamma_d\Delta_c)c^{-1}d^{-1}cd = (\lambda\Delta_c\Gamma_d)c^{-1}d^{-1}cd = (\lambda\Gamma_d)d^{-1}cd = (\lambda)cd = (\lambda\Delta_c)d = \lambda\Gamma_d\Delta_c$$

This proves our claim. Moreover, as p was chosen arbitrarily and independently of the definition of c and d, it means that the commutator $[c,d]$ fixes a prefix of each important point of a. In particular, this means that this commutator fixes all important points of α. □

To complete the setting of the scene for presenting our findings, we will quote the following theorem from B., Bleak and Matucci [4]. Most of the proof is implicit in Bleak and Salazar–Díaz [8], but the specific context is different.

Note that by a *free basis of rank n* we mean a set of n elements of V which generate a free subgroup of rank n.

Theorem 5.2.4. *Let $a, b \in V$ be non–torsion elements. Suppose that $\mathcal{I}_a \subseteq \mathcal{I}_b$. Then $\{a,b\}$ is not a free basis of rank 2.*

Proof. Let m and n be the smallest positive integers such that $c = a^m$ and $d = b^n$ have no non–trivial periodic orbits on Cantor space. Consider the commutator $\gamma = [c,d]$.

Note that if γ is torsion, then $\{c,d\}$ is not a free basis of rank two. In this case, $\{a,b\}$ cannot be a free basis of rank two either, as $\langle c,d\rangle \leq \langle a,b\rangle$, $c = a^m$ and $d = b^n$. Hence, we will assume that γ is not torsion. Let r be a minimal positive integer such that $\theta = \gamma^r$ admits no non–trivial periodic orbits on Cantor space.

Here we will emphasise that c and θ do not admit non–trivial periodic orbits in their action on Cantor space. Also, by Lemma 5.2.3, the commutator $\gamma = [c, d]$ fixes a prefix of each important point of a. Hence, $\theta = \gamma^r$ also fixes a prefix of each important point of a. Recall that $\mathcal{I}_a = \mathcal{I}_c$. This means that the support of θ is disjoint from the small neighbourhoods around important points of c. Therefore, by Lemma 4.4. of Bleak and Salazar–Díaz [8], we can find a non–zero integer s such that $Supp(\theta^{c^s}) \cap Supp(\theta) \cap Supp(c) = \emptyset$. This property can be interpreted as follows: for a power of c big enough, say c^s, all the support which θ is sharing with c can be moved entirely into the neighbourhoods of $\mathcal{I}_c$ by conjugating θ by c^s. As θ acts trivially in all of these neighbourhoods, we conclude that there are no points in Cantor space which can be in the support of all of the following three elements: c, θ, θ^{c^s}.

From here, setting $\mu = \nu = c$ and $\theta = \gamma^r$, we can follow the discussion of [8] from the point immediately following the proof of Lemma 4.4. throughout to the end of the paper. In this discussion, the element $\omega = [\theta^j, (\theta^{c^s})^j]$ is constructed, with carefully chosen positive integer j such that ω is torsion of order 1,2,3 or 6 by the analysis of the orbits of all possible types of points its support. Hence, $\langle \theta^{c^s}, \theta \rangle$ is not free of rank 2. This implies that $\langle c, \theta \rangle = \langle c, c^d \rangle$ is not free of rank 2. This, in turn, implies that $\langle c, d \rangle$ cannot be a free group of rank 2, and so finally, that $\langle a, b \rangle$ is not a free group of rank 2. This proves the theorem. □

5.3 On Dynamics of Free Products in V

We shall proceed to prove Theorem 5.1.1. The theorem could be summarised as follows: if a subgroup of V decomposes as a free product of two groups, and at least one of them is not torsion, then these groups admit Ping–Pong dynamics on a set of subsets of Cantor space.

Hence, we will assume that we have two non–trivial subgroups of V, A and B, which generate a free product. Without loss of generality, we will assume that there is an element $a \in A$ such that a is a non–torsion element. This means that the set of important points $\mathcal{I}_a$ is non–empty. There is also an induced action on the set X consisting of the full orbit of $\mathcal{I}_a$ under the group $\langle A, B \rangle$. We will then proceed with construction of subsets X_A and X_B. We will then prove that the sets X_A and X_B satisfy the criteria from Definition 5.2.1. This will prove that A and B admit Ping–Pong dynamics on X, and hence prove Theorem 5.1.1.

Note that we do not need an assumption about A and B generating a free product for the following lemma:

Lemma 5.3.1. *Let A and B be two subgroups of V. Let a be an infinite order element of A. Consider the sets X, X_A and X_B:*

$$X = \{(\mathcal{I}_a) \cdot w \mid w \in \langle A, B \rangle\}$$

$$X_A = \bigcup_{i=0}^{\infty} \left\{(\mathcal{I}_a) \cdot \textstyle\prod_{j=1}^{i} b_j a_j \mid b_1, \ldots, b_j \in B \setminus \{e\},\ a_1, \ldots, a_j \in A \setminus \{e\}\right\}$$

$$X_B = \bigcup_{i=0}^{\infty} \left\{(\mathcal{I}_a) \cdot b_0 \textstyle\prod_{j=1}^{i} a_j b_j \mid a_1, \ldots, a_j \in A \setminus \{e\},\ b_0, \ldots, b_j \in B \setminus \{e\}\right\}$$

Then the group $\langle A, B \rangle$ acts on the set X, and the sets X_A and X_B are as follows:

1. *$(X_A)b' \subseteq X_B$ for all $b' \in B \setminus \{e\}$;*
2. *$(X_B)a' \subseteq X_A$ for all $a' \in A \setminus \{e\}$.*

Proof. The group $\langle A, B \rangle$ acts on the set X as each element of the group represents a bijection on $\mathfrak{C}$.

Now, consider an element $S = (\mathcal{I}_a) \cdot b_0 \prod_{j=1}^{i} a_j b_j$ of the set X_B for some non–negative integer i and $a_1, \ldots, a_j \in A \setminus \{e\}$, $b_0, \ldots, b_j \in B \setminus \{e\}$. Let a' be such that $a' \in A \setminus \{e\}$. Consider the effect of a' on S:

$$(S)a' = ((\mathcal{I}_a) \cdot b_0 \prod_{j=1}^{i} a_j b_j)a' = (\mathcal{I}_a) \cdot \prod_{j=1}^{i+1} b'_j a'_j$$

for $b'_1 = b_0, b'_2 = b_1, \ldots, b'_{i+1} = b_i$ and $a'_1 = a_1, a'_2 = a_2, \ldots, a'_{i+1} = a'$. Therefore $(S)a' \in X_A$ as required.

Now consider an element $S = (\mathcal{I}_a) \cdot \prod_{j=1}^{i} b_j a_j$ of the set X_A for some non–negative integer i and $a_1, \ldots, a_j \in A \setminus \{e\}$, $b_1, \ldots, b_j \in B \setminus \{e\}$. Let b' be such that $b' \in B \setminus \{e\}$. Consider the effect of b' on S:

$$(S)b' = ((\mathcal{I}_a) \cdot \prod_{j=1}^{i} b_j a_j)b' = (\mathcal{I}_a) \cdot b_0 \prod_{j=1}^{i} a_j b'_j$$

for $b_0 = b_1, b'_1 = b_2, \ldots, b'_{i-1} = b_i, b'_i = b'$. Therefore $(S)b' \in X_B$ as required. □

Definition 5.3.2. Let A and B be non–trivial subgroups of V. Suppose that the group $\langle A, B\rangle$ naturally factors as the free product $A * B$. Let the sets X_A and X_B be defined as in Lemma 5.3.1. Then let us define the following subsets of the group $\langle A, B\rangle$:

$$T_A = \bigcup_{i=0}^{\infty} \{b_1 a_1 b_2 a_2 \dots b_i a_i \mid b_1, \dots, b_i \in B \setminus \{e\},\ a_1, \dots, a_i \in A \setminus \{e\}\}$$

$$T_B = \bigcup_{i=0}^{\infty} \{b_0 a_1 b_1 a_2 b_2 \dots a_i b_i \mid b_0, b_1, \dots, b_i \in B \setminus \{e\},\ a_1, \dots, a_i \in A \setminus \{e\}\}$$

$$T = T_A \cup T_B$$

Remark 5.3.3. Note that with T_A and T_B defined as above, we can more easily define sets X_A and X_B from Lemma 5.3.1:

$$X_A = \{(\mathcal{I}_a) \cdot w \mid w \in T_A\}$$
$$X_B = \{(\mathcal{I}_a) \cdot w \mid w \in T_B\}$$

Note also, that the union $T = T_A \cup T_B$ is in fact a *right transversal* for the cosets of the group A in the group $\langle A, B\rangle$, as by assumption $\langle A, B\rangle$ is canonically isomorphic to a free product.

Now, we only need to prove that if A and B generate a free product in V, then $X_B \setminus X_A \neq \emptyset$. Then the sets X_A and X_B will be the sets on which A and B admit a Ping–Pong.

Let us first prove a preliminary lemma:

Lemma 5.3.4. *Suppose that $a, w \in V$ such that $|a| = \infty$. Then $(\mathcal{I}_a)\cdot w = \mathcal{I}_{a^w}$.*

Proof. We will first show that $(\mathcal{I}_a) \cdot w \subseteq \mathcal{I}_{a^w}$.

Let $p \in \mathcal{R}_a$. Then $p = \lambda(\Gamma)^\infty$ for some words $\lambda, \Gamma \in \{0,1\}^*$ where there is a positive integer k such that $(\lambda\Gamma)a^k = \lambda$. Consider the image $p \cdot w$. As w acts as prefix substitution, there must be a positive integer s such that the following map is defined: $(\lambda(\Gamma)^s)w = \lambda'$ where $\lambda' \in \{0,1\}^*$. Then we also have also $p \cdot w = (\lambda(\Gamma)^\infty)w = \lambda'(\Gamma)^\infty$. Consider the effect of $(a^w)^k$ on the point $p \cdot w$:

$$[\lambda'(\Gamma)^\infty](a^w)^k = (\lambda(\Gamma)^\infty w)w^{-1}a^k w = [(\lambda(\Gamma)^\infty)a^k]w = (\lambda(\Gamma)^\infty)w = \lambda'(\Gamma)^\infty$$

Hence we know that the point $p \cdot w$ is fixed by $(a^k)^w$. Now let us consider

the effect of $(a^k)^w$ on the address $\lambda'\Gamma$ which is a prefix of $p \cdot w$:

$$(\lambda'\Gamma)(a^w)^k = (\lambda(\Gamma)^{s+1}w) \cdot w^{-1}a^k w = (\lambda(\Gamma)^s)w = \lambda'$$

Therefore the point $p \cdot w$ is a repelling point of a^w. This proves that $(\mathcal{R}_a) \cdot w \subseteq \mathcal{R}_{a^w}$.

Let $p \in \mathcal{A}_a$. Then $p = \lambda(\Gamma)^\infty$ for some words $\lambda, \Gamma \in \{0,1\}^*$ such that there is a positive integer k such that $(\lambda)a^k = \lambda\Gamma$. Consider the image $p \cdot w$. As w acts as prefix substitution, there must be a positive integer s such that the following map is defined: $(\lambda(\Gamma)^s)w = \lambda'$ where $\lambda' \in \{0,1\}^*$. Then we also have also $p \cdot w = (\lambda(\Gamma)^\infty)w = \lambda'(\Gamma)^\infty$. Consider the effect of $(a^w)^k$ on the point $p \cdot w$:

$$[\lambda'(\Gamma)^\infty](a^w)^k = (\lambda(\Gamma)^\infty w)w^{-1}a^k w = [(\lambda(\Gamma)^\infty)a^k]w = (\lambda(\Gamma)^\infty)w = \lambda'(\Gamma)^\infty$$

Hence we know that the point $p \cdot w$ is fixed by $(a^k)^w$. Now let us consider the effect of $(a^k)^w$ on the address λ' which is a prefix of $p \cdot w$:

$$(\lambda')(a^w)^k = (\lambda(\Gamma)^s w) \cdot w^{-1}a^k w = (\lambda(\Gamma)^{s+1})w = \lambda'\Gamma$$

Therefore the point $p \cdot w$ is an attracting point of a^w. This proves that $(\mathcal{A}_a) \cdot w \subseteq \mathcal{A}_{a^w}$.

As $(\mathcal{R}_a) \cdot w \subseteq \mathcal{R}_{a^w}$ and $(\mathcal{A}_a) \cdot w \subseteq \mathcal{A}_{a^w}$, we have $(\mathcal{I}_a) \cdot w \subseteq \mathcal{I}_{a^w}$.

Now we will show that $\mathcal{I}_{a^w} \subseteq (\mathcal{I}_a) \cdot w$. Note that as choices of a and w for the result above were arbitrary, we also have $(\mathcal{I}_{a^w}) \cdot w^{-1} \subseteq \mathcal{I}_a$. This implies that $(\mathcal{I}_{a^w}) \subseteq (\mathcal{I}_a) \cdot w$. Hence together with our previous result we have proven that $(\mathcal{I}_a) \cdot w = \mathcal{I}_{a^w}$. □

Using the above, we proceed to prove the statement $X_B \setminus X_A \neq \emptyset$. We will achieve it by first showing a considerably stronger property:

Lemma 5.3.5. *Let A, B, and T be defined as in Definition 5.3.2. Suppose that A contains an element a of infinite order. Then, if $\mathcal{I}_{a^{w_1}} = \mathcal{I}_{a^{w_2}}$ for some $w_1, w_2 \in T$, then $w_1 = w_2$.*

Proof. By Lemma 5.3.4 we have $\mathcal{I}_{a^{w_1}} = \mathcal{I}_a \cdot w_1$ and $\mathcal{I}_{a^{w_2}} = \mathcal{I}_a \cdot w_2$. Hence, $\mathcal{I}_{a^{w_1}} = \mathcal{I}_{a^{w_2}}$ is equivalent to $\mathcal{I}_a = (\mathcal{I}_a) \cdot w_1 w_2^{-1}$. Now, let us consider the following cases:

1. At least one of w_1, w_2 is the identity. Without loss of generality, let $w_2 = e$ and so the equation $\mathcal{I}_a = (\mathcal{I}_a) \cdot w_1 w_2^{-1}$ becomes $\mathcal{I}_a = (\mathcal{I}_a) \cdot w$

for some $w \in T$.

2. Not both of w_1, w_2 are in $T_A \setminus \{e\}$ and not both are in T_B. Without loss of generality, $w_1 \in T_A \setminus \{e\}$ and $w_2 \in T_B$. Hence, there are some positive integers j and k such that:

$$w_1 = b_1 a_1 \dots b_j a_j$$
$$w_2 = b_{j+k} a_{j+k-1} b_{j+k-1} \dots a_{j+1} b_{j+1}$$

for some $a_i \in A \setminus \{e\}$ and some $b_i \in B \setminus \{e\}$. Then:

$$w_1 w_2^{-1} = b_1 a_1 \dots b_j a_j b_{j+1}^{-1} a_{j+1}^{-1} \dots a_{j+k-1}^{-1} b_{j+k}^{-1} \in T_B \setminus B$$

Therefore, the equation becomes $\mathcal{I}_a = (\mathcal{I}_a) \cdot w$ for some $w \in T_B \setminus B$.

3. We have both $w_1, w_2 \in T_A \setminus \{e\}$ or both $w_1, w_2 \in T_B$. Without loss of generality, let w_1 be longer or of the same length as w_2. The word $w_1 w_2^{-1}$ might admit cancellations in the middle, at most as many as the length of the word w_2. Now, if $w_1 = w_2$, then there is nothing to prove. Hence we will presume that $w_1 \neq w_2$ and so $w_1 w_2^{-1}$ is a non–trivial word which starts with a non–trivial element of B, and may finish with a non–trivial word from A or B. Hence, $w_1 w_2^{-1} \in T$.

Thus, the equation becomes $\mathcal{I}_a = (\mathcal{I}_a) \cdot w$ for some $w \in T$.

Hence we will instead prove that if $\mathcal{I}_a = \mathcal{I}_{a^w}$ for some $w \in T$, then $w = e$. Note that this equation is equivalent to $\mathcal{I}_a = \mathcal{I}_{a^{w^{-1}}}$ for some $w \in T$. By Theorem 5.2.4 we conclude that $\{a, a^{w^{-1}}\}$ cannot be a free basis of rank 2. However, recall that the groups A and B canonically form a free product.

Then, if $w \in T_B$, namely it starts and finishes with a non-trivial element of B, then each non—trivial power of the generator $a^{w^{-1}}$, given by $(a^{w^{-1}})^m = w a^m w^{-1}$, admits no cancellations with the letter a on the left or right. Hence, $\{a, a^{w^{-1}}\}$ must be a free basis of rank 2, which gives us a contradiction.

Thus, presume that $w \in T_A \setminus \{e\}$. Hence suppose that $w = w'a'$ for some words $w' \in T_B$ and $a' \in A \setminus \{e\}$. But then the word $(a^{w^{-1}})^m = ((a^{a'^{-1}})^{w'^{-1}})^m = w'(a^{a'^{-1}})^m w'^{-1}$ admits no cancellations with the letter a on the left or right. Also, $(a^{a'^{-1}})^m$ remains an element of infinite order,

and so again, $\{a, a^{w^{-1}}\}$ must be a free basis of rank 2, which gives us a contradiction.

Therefore, we must have $w = e$ which gives us our desired conclusion. □

Corollary 5.3.6. *For the the groups A and B as in Definition 5.3.2, and the sets X_A and X_B as in Lemma 5.3.1, we have $X_A \cap X_B = \emptyset$.*

Proof. By Lemma 5.3.4 and Lemma 5.3.5. □

Corollary 5.3.7. *Let A and B be subgroups of V and $a \in A$ have infinite order. Let the map f be as defined in Theorem 5.1.1. Suppose that f is an isomorphism (i.e. A and B generate a free product). Let the sets X_A and X_B be as in Lemma 5.3.1. Then $X_B \setminus X_A \neq \emptyset$.*

Proof. Consider $S = (\mathcal{I}_a) \cdot b$ for some $b \in B \setminus \{e\}$. By definition, $S \in X_B$. However, by Corollary 5.3.6 the sets X_A and X_B are disjoint, and so $S \in X_B \setminus X_A$. □

Finally, this last corollary in the same time proves Theorem 5.1.1, which is the central result of this chapter:

Corollary 5.3.8. *Let A and B be subgroups of V and $a \in A$ have infinite order. Let the map f be as defined in Theorem 5.1.1. Suppose that f is an isomorphism. Let the sets X_A and X_B be as defined in Lemma 5.3.1. Then the groups A and B admit a Ping–Pong action on the sets X_A and X_B.*

This proves Theorem 5.1.1.

Proof. By Lemma 5.3.1 and Corollary 5.3.7 the sets X_A and X_B satisfy conditions from Definition 5.2.1. □

Bibliography

[1] Stefan Banach and Alfred Tarski (1924), *Sur la décomposition des ensembles de points en parties respectivement congruentes*, Fund. Math. 6, 244–277.

[2] Daniel Bennett and Collin Bleak (2016), *A dynamical definition of f.g. virtually free groups*, Int. J. Algebra Comput. 26, 105–121.

[3] Ewa Bieniecka, *On constructing elements with a prescribed centraliser in the Thompson's Group V*, (in preparation).

[4] Ewa Bieniecka, Collin Bleak and Francesco Matucci, *Free subgroups of Thompson's group V*, (in preparation).

[5] Collin Bleak, Hannah Bowman, Alison Gordon Lynch, Garrett Graham, Jacob Hughes, Francesco Matucci and Eugenia Sapir (2013), *Centralizers in the R. Thompson group V_n*, Groups Geom. Dyn. 7, no. 4, 821–865.

[6] Collin Bleak and Roman Kogan, *vTrees*, 2007 (updated 2015).

[7] Collin Bleak , Francesco Matucci and Max Neunhöffer (2016), *Embeddings into Thompson's group V and coCF groups*, Journal of the London Mathematical Society, vol 94, no. 2, 583–597.

[8] Collin Bleak and Olga Salazar–Díaz (2013), *Free products in R. Thompson's group V*, Trans. Amer. Math. Soc. 365, no. 11, 5967–5997.

[9] Collin Bleak and Martyn Quick (2017), *The infinite simple group V of Richard J. Thompson: presentations by permutations*, Groups Geom. Dyn. (to appear).

[10] Matthew G. Brin (1997), *The chameleon groups of Richard J. Thompson: automorphisms and dynamics*, Inst. Hautes Etudes Sci. Publ. Math. (1996), no. 84, 5–33.

[11] Matthew G. Brin (2004), *Higher dimensional Thompson groups*, Geom. Dedicata 108, 163–192.

[12] Matthew G. Brin and Craig C. Squier (1985), *Groups of piecewise linear homeomorphisms of the real line*, Invent. Math. 79, 485–498.

[13] Kenneth S. Brown (1987), *Finiteness properties of groups*, J. Pure Appl. Algebra 44, 45–75.

[14] J. W. Cannon, W. J. Floyd and W. R. Parry (1996), *Introductory Notes on Richard Thompson's Groups*, L'Enseign. Math. 42, 215–256.

[15] Robert Fricke and Felix Klein (1897), *Vorlesungen über die theorie der automorphen funktionen*, vol. 1, Teubner, Leipzig.

[16] Pierre de la Harpe (2000), *Topics in Geometric Group Theory*, University of Chicago Press.

[17] Graham Higman (1974), *Finitely presented infinite simple groups*, Department of Pure Mathematics, Department of Mathematics, I.A.S. Australian National University, Canberra, 1974, Notes on Pure Mathematics, No. 8.

[18] Derek F. Holt and Claas E. Röver (2006), *Groups with indexed co-word problem*, Internat. J. Algebra Comput. 16, no. 5, 985–1014.

[19] Martin Kassabov and Francesco Matucci (2012), *The simultaneous conjugacy problem in groups of piecewise linear functions*, Groups Geom. Dyn.,6, 279-315.

[20] Felix Klein (1883), *Neue Beiträge zur Riemann'schen Functionentheorie*, Math. Annalen 21, 141–218.

[21] Jörg Lehnert (2008), *Gruppen von quasi-Automorphismen*, PhD thesis, Goethe Universität, Frankfurt.

[22] Jörg Lehnert and Pascal Schweitzer (2007), *The co–word problem for the Higman—Thompson group is context-free*, Bull. Lond. Math. Soc. 39, no. 2, 235–241.

[23] Yash Lodha and Justin Tatch Moore (2016), *A nonamenable finitely presented group of piecewise projective homeomorphisms*, Groups Geom. Dyn. 10, 177–200.

[24] Francesco Matucci (2008), *Algorithms and classification in groups of piecewise-linear homeomorphisms*, PhD Thesis, Cornell University, Ithaca NY.

[25] Nicolas Monod (2013), *Groups of piecewise projective homeomorphisms*, Proceedings of the National Academy of Sciences of the United States of America, 110, no. 12, 4524–4527.

[26] John von Neumann (1929), *Zur allgemeinen Theorie des Maßes*, Fund. Math. 13, 73-–116.

[27] Alexander Ol′shanskiĭ (1980), *On the question of the existence of an invariant mean on a group*, Uspekhi Mat. Nauk 35, no. 4, 199–200.

[28] Olga Patricia Salazar–Díaz (2010), *Thompson's group V from a dynamical viewpoint*, Internat. J. Algebra Comput., 20, no. 1, 39–70.

[29] Alfred Tarski (1929), *Sur les fonctions additives dans les classes abstraites et leur application au problème de la mesure*, C. R. Soc. Sc. Varsovie 22, 114–117.

[30] Alfred Tarski (1938), *Algebraische Fassung des Maßproblems*, Fund. Math. 31, 47–66.

[31] Richard J. Thompson (1965), Handwritten widely circulated notes.

[32] Samuel White (1988), *The group generated by* $x \mapsto x+1$ *and* $x \mapsto x^p$ *is free*, J. Algebra 118, 408–422.